GigaDevice Semiconductor Inc.

GD32E50x
Arm® Cortex®-M33 32-bit MCU

For GD32E503xx, GD32E505xx, GD32E507xx, GD32E508xx,

GD32EPRTxx

User Manual

Revision 2.0

(Dec. 2024)

Table of Contents

List of Figures

List of Tables

1. System and memory architecture

The devices of GD32E50x series are 32-bit general-purpose microcontrollers based on the Arm® Cortex ®-M33 processor. The Arm® Cortex®-M33 processor includes two AHB buses known as Code and System buses. All memory accesses of the Arm® Cortex®-M33 processor are executed on these two 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.

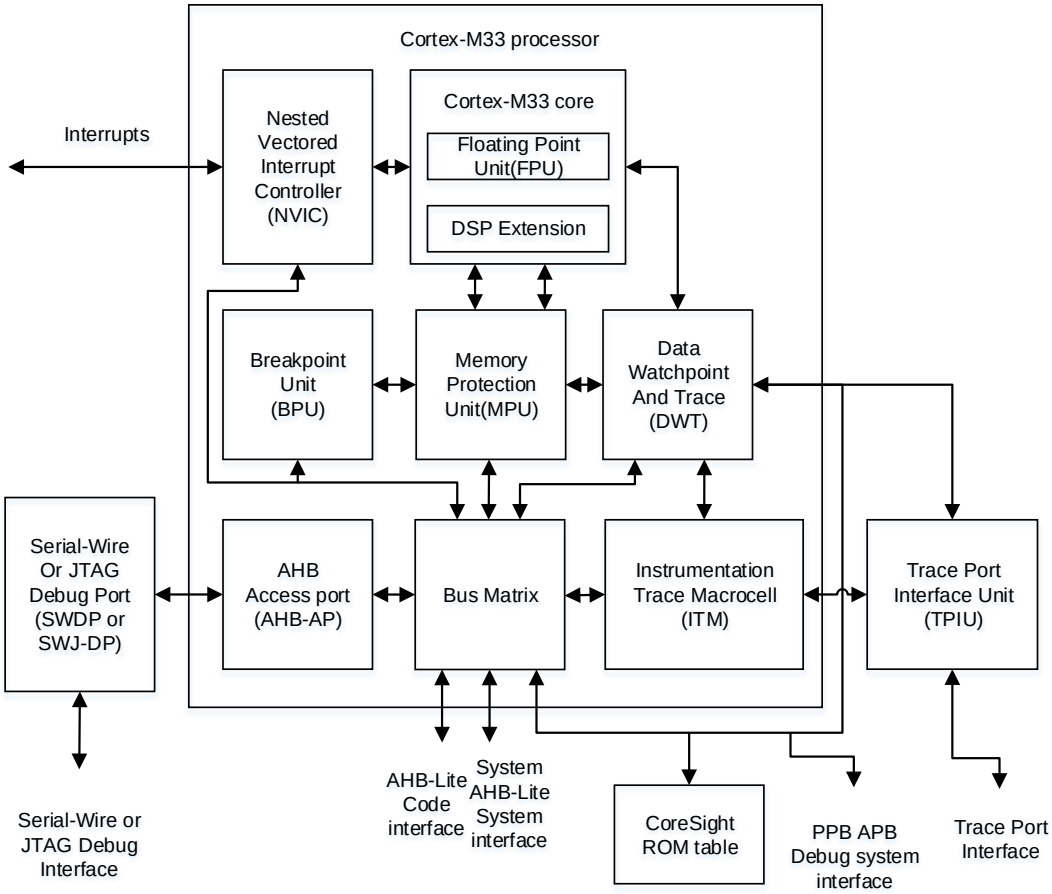
1.1. Arm® Cortex®-M33 processor

The Cortex®-M33 processor is a 32-bit processor that possesses low interrupt latency and low-cost debug. The characteristics of integrated and advanced make the Cortex®-M33 processor suitable for market products that require microcontrollers with high performance and low power consumption. The Cortex®-M33 processor is based on the Armv8 architecture and supports a powerful and scalable instruction set including general data processing I/O control tasks, advanced data processing bit field manipulations and DSP. Some system peripherals listed below are also provided by Cortex®-M33:

- Internal Bus Matrix connected with Code bus, System bus, and Private Peripheral Bus (PPB) and debug accesses
- Nested Vectored Interrupt Controller (NVIC)
- Breakpoint Unit (BPU)
- Data Watchpoint and Trace (DWT)
- Instrumentation Trace Macrocell (ITM)
- Serial Wire JTAG Debug Port (SWJ-DP)
- Trace Port Interface Unit (TPIU)
- Memory Protection Unit (MPU)
- Floating Point Unit (FPU)
- DSP Extension (DSP)

Figure 1-1. The structure of the Cortex®-M33 processor shows the Cortex®-M33 processor block diagram. For more information, please refer to the Arm® Cortex®-M33 Technical Reference Manual.

Figure 1-1. The structure of the Cortex®-M33 processor



1.2. System architecture

A 32-bit multilayer bus is implemented in the GD32E50x 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

	CBUS	SBUS	DMA0	DMA1	ENET	OTGHS
FMC	1		1	1		1
SRAM	1	1	1	1	1	1
AHB1		1	1	1		
AHB2	1	1	1	1	1	1

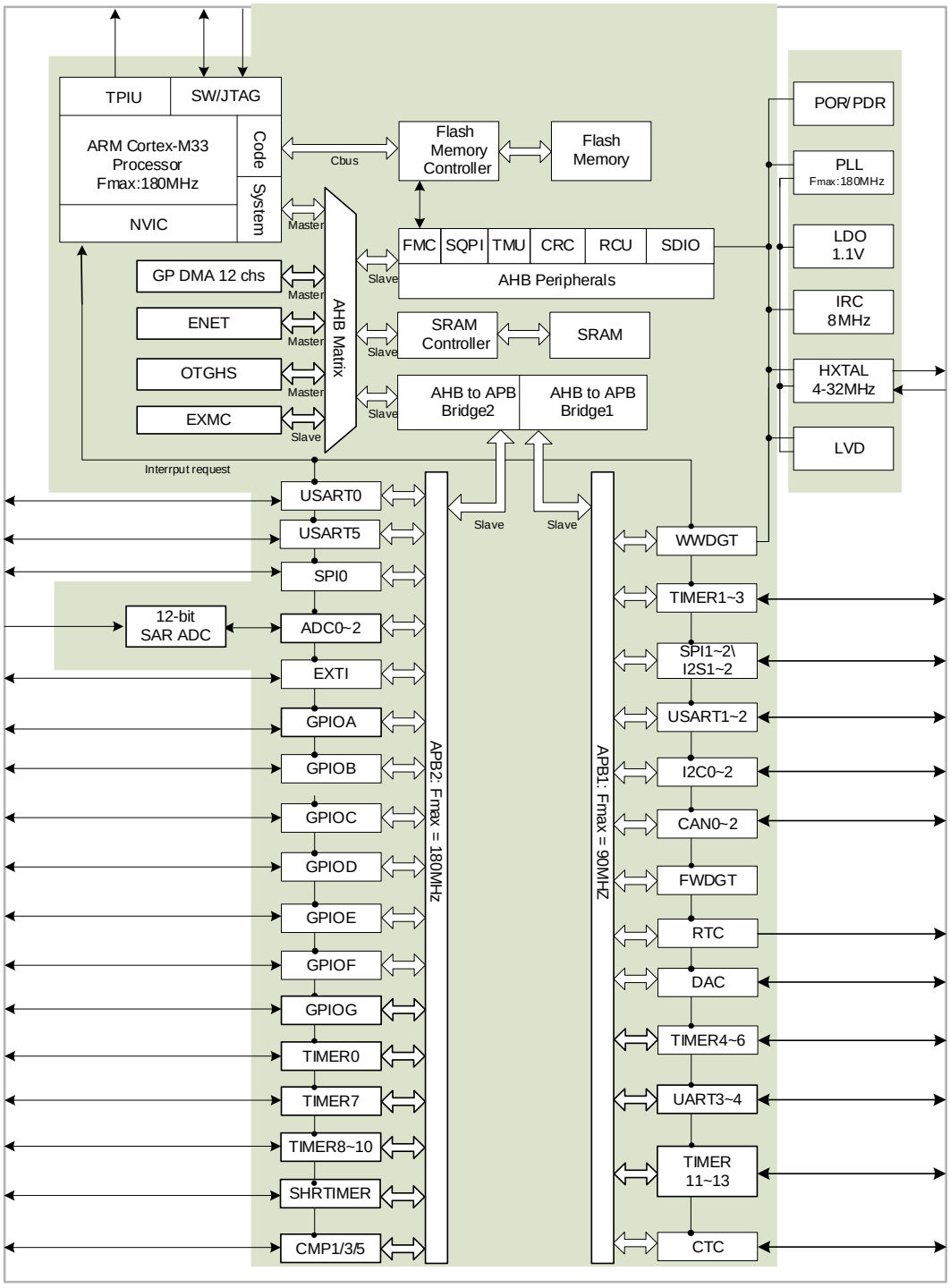
	CBUS	SBUS	DMA0	DMA1	ENET	OTGHS
APB1		1	1	1		
APB2		1	1	1		

As is shown above, there are several masters connected with the AHB interconnect matrix, including CBUS, SBUS, DMA0, DMA1, ENET and OTGHS. CBUS is the code bus of the Cortex®-M33 core, which is used for any instruction fetch and data access to the Code region. Similarly, SBUS is the system bus of the Cortex®-M33 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. OTGHS is the high speed USB 2.0 module which supports OTG protocol.

There are also several slaves connected with the AHB interconnect matrix, including FMC, SRAM, AHB1, AHB2, APB1 and APB2. FMC is the bus interface of the flash memory controller. SRAM is on-chip static random access memories. AHB1 is the AHB bus connected with all of the AHB slaves except EXMC and SQPI. AHB2 is the AHB bus connected with EXMC and SQPI. 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 90 MHz, APB2 operates at full speed (up to 180MHz depending on the device).

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

Figure 1-2. GD32E50x system architecture



1.3. Memory map

The Arm® Cortex®-M33 processor is structured in Harvard architecture which can use separate buses to fetch instructions and load/store data. 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®-M33 since the bus address width is 32-bit.

Additionally, a pre-defined memory map is provided by the Cortex®-M33 processor to reduce the software complexity of repeated implementation of different device vendors. In the map, some regions are used by the Arm® Cortex®-M33 system peripherals which can not be modified. However, the other regions are available to the vendors. ***Table 1-2. Memory map of GD32E50x devices*** shows the memory map of the GD32E50x 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 GD32E50x devices

Pre-defined Regions	Bus	Address	Peripherals
External device	AHB3	0xC000 0000 - 0xDFFF FFFF	Reserved
		0xB000 0000 - 0xBFFF FFFF	SQPI_PSRAM(MEM)
		0xA000 1400 - 0xAFFF FFFF	Reserved
		0xA000 1000 - 0xA000 13FF	SQPI_PSRAM(REG)
		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	AHB1	0x5000 0000 - 0x5003 FFFF	USBHS
		0x4008 0400 - 0x4FFF FFFF	Reserved
		0x4008 0000 - 0x4008 03FF	TMU
		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

Pre-defined Regions	Bus	Address	Peripherals
		0x4002 1400 - 0x4002 17FF	Reserved
		0x4002 1000 - 0x4002 13FF	RCU
		0x4002 0C00 - 0x4002 0FFF	Reserved
		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	CMP
		0x4001 7800 - 0x4001 7BFF	Reserved
		0x4001 7400 - 0x4001 77FF	SHRTIMER
		0x4001 7000 - 0x4001 73FF	USART5
		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 D000 - 0x4000 FFFF	Reserved
		0x4000 CC00 - 0x4000 CFFF	CAN2

Pre-defined Regions	Bus	Address	Peripherals
		0x4000 C800 - 0x4000 CBFF	CTC
		0x4000 C400 - 0x4000 C7FF	Reserved
		0x4000 C000 - 0x4000 C3FF	I2C2
		0x4000 8C00 - 0x4000 BFFF	Reserved
		0x4000 8800 - 0x4000 8BFF	CAN2SRAM
		0x4000 8400 - 0x4000 87FF	USBSRAM_B
		0x4000 8000 - 0x4000 BFFF	Reserved
		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	I2S2_add
		0x4000 3C00 - 0x4000 3FFF	SPI2/I2S2
		0x4000 3800 - 0x4000 3BFF	SPI1/I2S1
		0x4000 3400 - 0x4000 37FF	I2S1_add
		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

Pre-defined Regions	Bus	Address	Peripherals
SRAM	AHB	0x2007 0000 - 0x3FFF FFFF	Reserved
		0x2006 0000 - 0x2006 FFFF	Reserved
		0x2003 0000 - 0x2005 FFFF	Reserved
		0x2002 0000 - 0x2002 FFFF	Reserved
		0x2000 0000 - 0x2001FFFF	SRAM
Code	AHB	0x1FFF F810 - 0x1FFF FFFF	Reserved
		0x1FFF F800 - 0x1FFF F80F	Option Bytes
		0x1FFF F000 - 0x1FFF F7FF	Boot loader
		0x1FFF C010 - 0x1FFF EFFF	
		0x1FFF C000 - 0x1FFF C00F	
		0x1FFF B000 - 0x1FFF BFFF	
		0x1FFF 7800 - 0x1FFF AFFF	Reserved
		0x1FFF 7000 - 0x1FFF 77FF	OTP
		0x1FFF 0000 - 0x1FFF 6FFF	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
		0x0808 0000 - 0x082F FFFF	Reserved
		0x0800 0000 - 0x0807 FFFF	Main Flash
		0x0030 0000 - 0x07FF FFFF	Reserved
		0x0010 0000 - 0x002F FFFF	Reserved
		0x0008 0000 - 0x000F FFFF	Reserved
		0x0002 0000 - 0x0007 FFFF	Aliased to Main Flash or Boot loader
		0x0000 0000 - 0x0001 FFFF	

1.3.1. On-chip SRAM memory

The GD32E50x series of devices contain up to 128 KB of on-chip SRAM which starts at the address 0x2000 0000. It supports byte, half-word (16 bits), and word (32 bits) accesses.

1.3.2. On-chip flash memory overview

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

- Up to 512KB of main flash memory.
- Up to 18KB of information blocks for the boot loader.
- Option bytes to configure the device.

GD32E503xx microcontrollers where the flash memory density ranges between 256 and 512

Kbytes are called High-density devices (GD32E50x_HD).

GD32E505xx and GD32E507xx microcontrollers are called connectivity line devices (GD32E50x_CL).

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

1.4. Boot configuration

The GD32E50x 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

After power-on sequence or a system reset, the Arm® Cortex®-M33 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 (HD series original memory space beginning at 0x1FFF E000, please refer to ***Table 2-1. GD32E50x_HD and GD32E50x_CL base address and size for flash memory*** for other series addresses) 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 bootloader is located in the system memory, which is used to reprogram the Flash memory. The boot loader can be activated through the following serial interfaces: USART0 or USART1 or USB.

1.5. Device electronic signature

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: 0x0080 indicates 128 Kbytes.
15:0	FLASH_DENSITY [15:0]	Flash memory density The value indicates the Flash memory density of the device in Kbytes. Example: 0x0200 indicates 512 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. Flash memory controller (FMC)

2.1. Overview

The flash memory controller, FMC, provides all the necessary functions for the on-chip flash memory. A little waiting time is needed while CPU executes instructions stored from the 512K bytes of the flash. It also provides page erase, mass erase, and program operations for flash memory.

2.2. Characteristics

- Up to 512KB of on-chip flash memory for instruction and data.
- 0~4 waiting time within 512KB bytes when CPU executes instructions.
- Pre-fetch buffer to speed read operations.
- I-cache with 2K bytes which organized as 64 cache line of 2 X 128 bits.
- D-cache with 256 bytes which organized as 8 cache line of 2 X 128 bits.
- The flash page size is 8KB.
- Word programming, page erase and mass erase operation.
- 2KB OTP(one-time program) block used for user data storage.
- 16B option bytes block for user application requirements.
- Option bytes are uploaded to the option byte control registers when the system is 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

The page size is 8KB. Each page can be erased individually.

The following table shows the details of flash organization.

Table 2-1. GD32E50x_HD and GD32E50x_CL base address and size for flash memory

Block	Name	Address range	Size(bytes)
Main flash block	Page 0	0x0800 0000 - 0x0800 1FFF	8KB
	Page 1	0x0800 2000 - 0x0800 3FFF	8KB
	Page 2	0x0800 4000 - 0x0800 5FFF	8KB
	Page 63	0x0807 E000 - 0x0807 FFFF	8KB

Block		Name	Address range	Size(bytes)
Information block	GD32E50x_HD	Boot loader area	0x1FFF E000- 0x1FFF F7FF	6KB
	GD32E50x_CL		0x1FFF B000- 0x1FFF F7FF	18KB
Option bytes block		Option bytes	0x1FFF F800 - 0x1FFF F80F	16B
One-time program block		OTP bytes	0x1FFF_7000~0x1FFF_77FF	2KB

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 AHB BUS from the CPU.

Wait state added:

The WSCNT bits in the FMC_WS register needs to be configured correctly depend on the AHB clock frequency when reading the flash memory. The relation between WSCNT and AHB clock frequency is show as the ***Table 2-2. The relation between WSCNT and AHB clock frequency***.

Table 2-2. The relation between WSCNT and AHB clock frequency

AHB clock frequency	WSCNT configured
<= 36MHz	0 (0 wait state added)
<= 72MHz	1 (1 wait state added)
<= 108MHz	2 (2 wait state added)
<= 144MHz	3 (3 wait state added)
<= 180MHz	4 (4 wait state added)

If system reset occurs, the AHB clock frequency is 8MHz and the WSCNT is 0.

Note:

1. If want to increase the AHB clock frequency. First, refer to the correspondence table between WSCNT bit and AHB clock frequency, configure the WSCNT bits according to the target AHB clock frequency. Then, increase the AHB clock frequency to the target frequency. It is forbidden to increase the AHB clock frequency before configure the WSCNT.

2. If want to decrease the AHB clock frequency. First, decrease the AHB clock frequency to target frequency. Then refer to the correspondence table between WSCNT bit and AHB clock frequency, configure the WSCNT bits according the target AHB clock frequency. It is forbidden to configure the WSCNT bits before decrease the AHB clock frequency.

Because the wait state is added, the read efficiency is very low (such as, add 4 wait state when 180MHz). In order to speed up the read access, there are some functions performed.

Current buffer:

The current buffer is always enabled. Each time read from flash memory, 128-bit data get and store in current buffer. The CPU only need 32-bit or 16-bit in each read operation. So in the case of sequential code, the next data can get from current buffer without repeat fetch from flash memory.

Pre-fetch buffer:

The pre-fetch buffer is enabled by set the PFEN bit in the FMC_WS register. The pre-fetch buffer is only performed only when fetching instructions. In the case of sequential code, when CPU execute the current buffer data (128-bit), it takes at least 4 clocks for 32-bit operation and at least 8 clocks for 16-bit operation. In this case, pre-fetch the data of next double-word address from flash memory and store to pre-fetch buffer. So when the CPU finish the current buffer and need execute the next data, the pre-fetch buffer hit.

I-cache:

I-cache is enabled by set the ICEN bit in the FMC_WS register. The I-cache is only used when fetching instructions. The I-cache have 2K bytes which organized as 64 cache lines, each cache lines is 2 X 128bits.

If the instructions data is in I-cache (I-cache hit), the CPU read data from I-cache without any wait state. If the instructions data is not in I-cache (I-cache miss) and not in current buffer/pre-fetch buffer, the cache line fetch from flash memory and copied to I-cache. If all cache line filled, LRU (least recently used) policy used to replace the cache line.

D-cache:

D-cache is enabled by set the DCEN bit in the FMC_WS register. The D-cache is only used when fetching data by CPU (not by DMA). And the option byte is not cacheable. The D- cache have 256 bytes which organized as 8 cache lines, each cache lines is 2 X 128bits.

If the data is in D-cache (D-cache hit), the CPU read data from D-cache without any wait state. If the data is not in D-cache (D-cache miss) and not in current buffer, the cache line fetch from flash memory and copied to D-cache. If all cache line filled, LRU (least recently used) policy used to replace the cache line.

2.3.3. Unlock the FMC_CTL register

After reset, the FMC_CTL register is not accessible in write mode, and the LK bit in the FMC_CTL register is reset to 1. An unlocking sequence consists of two write operations to the FMC_KEY register to open the access to the FMC_CTL register. The two write operations are writing 0x45670123 and 0xCDEF89AB to the FMC_KEY register. After the two write operations, the LK bit in the FMC_CTL register is reset to 0 by hardware. The software can lock the FMC_CTL again by setting the LK bit in the FMC_CTL register to 1. Any wrong

operations to the FMC_KEY, will set the LK bit to 1, and lock the FMC_CTL register, and lead to a bus error.

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

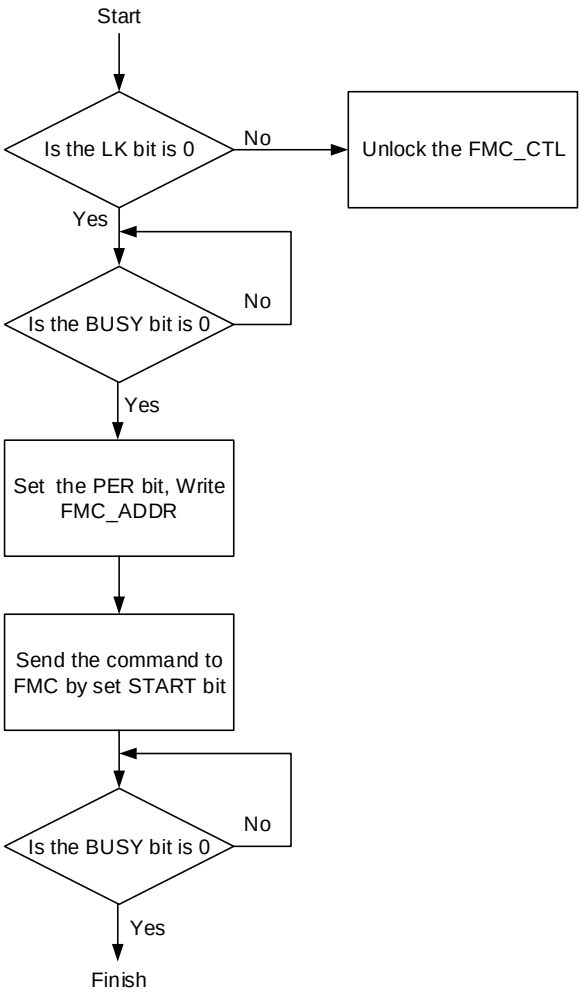
2.3.4. Page erase

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_CTL register if necessary.
- Check the BUSY bit in the FMC_STAT register 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 the FMC_CTL register.
- Write the page absolute address (0x08XX XXXX) into the FMC_ADDR registers.
- Send the page erase command to the FMC by setting the START bit in the FMC_CTL register.
- Wait until all the operations have finished by checking the value of the BUSY bit in the FMC_STAT register.
- Read and verify the page if required.

When the operation is executed successfully, the ENDF bit in the FMC_STAT register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set. Note that a correct target page address must be confirmed. Otherwise, the software may run out of control if the target erase page is being used to fetch codes or access data. The FMC will not provide any notification when that happens. 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_CTL register is set. The software can check the WPERR bit in the FMC_STAT register 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



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 entire flash block by setting the MER bit to 1 in the FMC_CTL register. The following steps show the mass erase register access sequence.

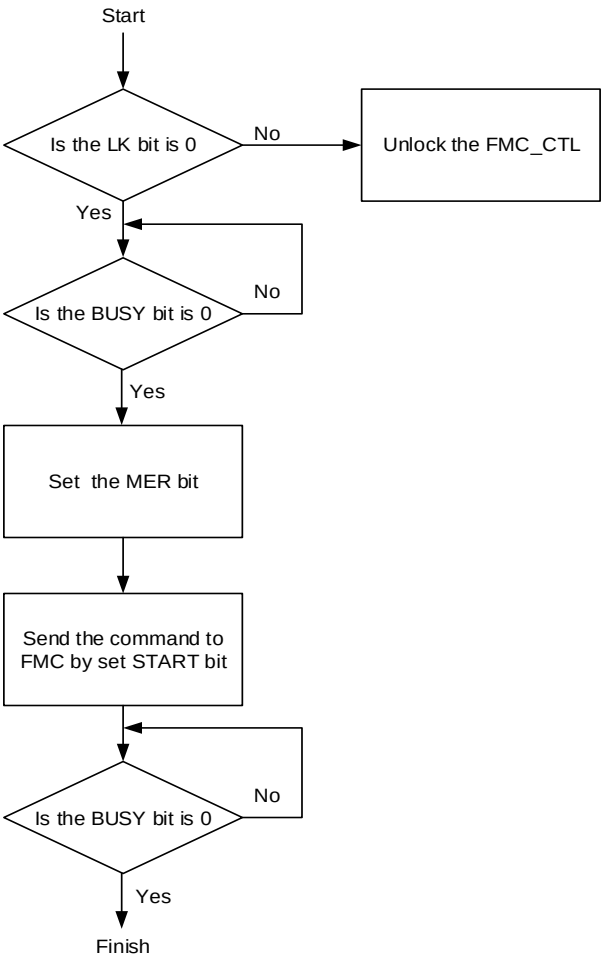
- Unlock the FMC_CTL register if necessary.
- Check the BUSY bit in the FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
- Set the MER bit in the FMC_CTL register if erase entire flash.
- Send the mass erase command to the FMC by setting the START bit in the FMC_CTL register.
- Wait until all the operations have been finished by checking the value of the BUSY bit in the FMC_STAT register.
- Read and verify the flash memory if required.

When the operation is executed successfully, the ENDF bit in the FMC_STAT register is set,

and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register 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 using the debugging tool that accesses the FMC registers directly. Additionally, the mass erase operation will be ignored if any page is erase/program protected. In this condition, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTL register is set. The software can check the WPERR bit in the FMC_STAT register to detect this condition in the interrupt handler.

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. While actually, the data program to flash memory is 32-bits.

The following steps show the register access sequence of the programming operation.

- Unlock the FMC_CTL register if necessary.
- Check the BUSY bit in the FMC_STAT register 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 the FMC_CTL register.
- Write the data to be programed with desired absolute address (0x08XX XXXX).
  If program is 32-bit word, the write once and the data program to flash memory. The data to be programed must word alignment.
  If program is 16-bit, the write twice to form a 32-bit data and then the 32-bit data program to flash memory. The data to be programed must word alignment.
  For less program time, suggest the program use 32-bit.
- Wait until all the operations have been finished by checking the value of the BUSY bit in the FMC_STAT register.
- Read and verify the flash memory if required.

When the operation is executed successfully, the ENDF bit in the FMC_STAT register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set. Note that there are some program error need caution:

The programming operation checks the address if it has been erased or not. If the address has not been erased, the PGERR bit in the FMC_STAT register will be set. Each word can be programmed only one time after erase and before next erase 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 the WPERR bit in the FMC_STAT will be set.

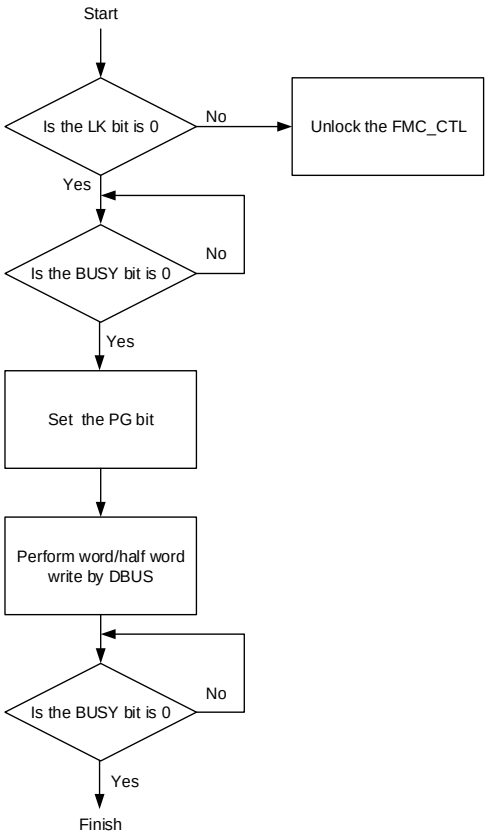
In the following cases, the PGAERR bit in the FMC_STAT register will be set.

- The program use byte write (not 32-bit or 16-bit write)
- The program size is not equal previous size. It not allow mix 32-bit with 16-bit write.
- The write is not alignment.

Note: If the program is not write total 32bits, the data is not program to the flash memory without any notice.

In these conditions, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTL register is set. The software can check the PGERR bit, PGAERR bit or WPERR bit in the FMC_STAT register to detect which condition occurred in the interrupt handler. The following figure shows the word programming operation flow.

Figure 2-3. Process of word program operation



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

2.3.7. OTP programming

The OTP programming method is same as the main flash programming. The OTP block can only be programed once and cannot be erased.

Note: It must ensure the OTP programming sequence completely without any unexpected interrupt, such as system reset or power down. If unexpected interrupt occurs, there is very little probability of corrupt the data stored in flash memory.

2.3.8. 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_CTL register if necessary.
- Check the BUSY bit in the FMC_STAT 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 the FMC_CTL register if necessary.
- Wait until the OBWEN bit is set in the FMC_CTL register.
- Set the OBER bit in the FMC_CTL register.
- Send the option bytes erase command to the FMC by setting the START bit in the FMC_CTL register.
- Wait until all the operations have been finished by checking the value of the BUSY bit in the FMC_STAT register.
- Read and verify the flash memory if required.

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

2.3.9. Option bytes modify

The FMC provides an erase / program function which is used to modify the option bytes block in flash. There are 8 pairs of option bytes. The MSB is the complement of the LSB in each pair. 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_CTL register if necessary.
- Check the BUSY bit in the FMC_STAT 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 the FMC_CTL register if necessary.
- Wait until the OBWEN bit is set in the FMC_CTL register.
- Set the OBPG bit in the FMC_CTL register.
- A 32-bit word/16-bit half word write at desired address. The write method is similar to main flash programming.
- Wait until all the operations have been finished by checking the value of the BUSY bit in the FMC_STAT register.
- Read and verify the flash memory if required access.

When the operation is executed successfully, the ENDF bit in the FMC_STAT register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set. Note that there programming errors may occur. The PGERR bit and PGAERR bit can be set which is similar to main flash programming.

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

2.3.10. Option bytes description

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

Table 2-3. Option bytes

Address	Name	Description
0x1fff f800	SPC	option bytes security protection value 0xA5 : no security protection any value except 0xA5 or 0xCC : protection level low 0xCC : protection level high
0x1fff f801	SPC_N	SPC complement value
0x1fff f802	USER	[7:3]: reserved [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 8KB flash protection. These bits totally controls the first 248KB 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

2.3.11. 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_STAT register will 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, then a system reset is necessary.

Table 2-4. OB_WP bit for pages protected

OB_WP bit	pages protected
OB_WP[0]	page 0
OB_WP[1]	page 1
OB_WP[2]	page 2
. . .	. . .
OB_WP[30]	page 30
OB_WP[31]	page 31 ~ page 63

2.3.12. Security protection

The FMC provides a security protection function to prevent illegal code/data access to 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.

Protection level low: when setting SPC byte value to any value except 0xA5 or 0xCC, the low security protection is performed. Note that a power reset should be followed instead of a system reset if the SPC modification has been performed while the debug module is still connected to JTAG/SWD device. Under the low security protection, the main flash can only be accessed by user code and the first 8KB flash is under erase/program protection. In debug mode, boot from SRAM or boot loader mode, all operations to main flash is forbidden. If a read operation to main flash in debug mode, boot from SRAM or 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 the FMC_STAT register will be set. Option bytes block are accessible by all operations, which can be used to disable the security protection. Back to no protection level by setting SPC byte and its complement value to 0x5AA5, then a mass erase for main flash will be performed.

Protection level high: when setting SPC byte to 0xCC, protection level high performed. When this level is programmed, debug mode, boot from SRAM or boot from boot loader mode are disabled. The main flash block is accessible by all operations from user code. The SPC byte cannot be reprogrammed. So, if protection level high is programmed, it cannot move back to protection level low or no protection level.

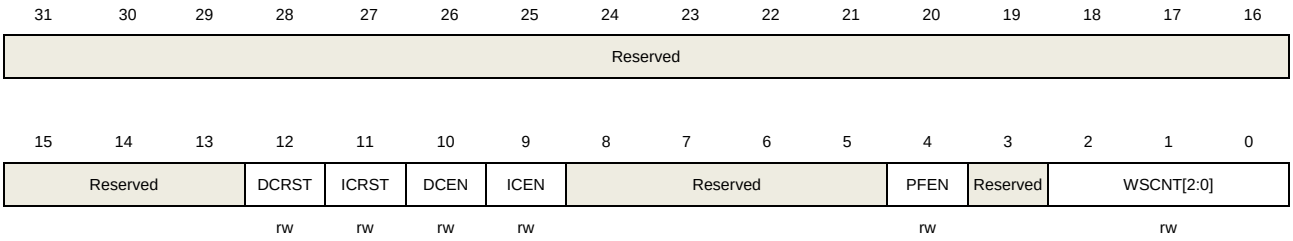
2.4. Register definition

FMC base address: 0x4002 2000

2.4.1. Wait state register (FMC_WS)

Address offset: 0x00
Reset value: 0x0000 0630

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



Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12	DCRST	D-cache reset. This bit can be write only when DCEN is set to 0. 0: No effect 1: D-cache reset
11	ICRST	I-cache reset. This bit can be write only when ICEN is set to 0. 0: No effect 1: I-cache reset
10	DCEN	D-cache enable 0: D-cache disable 1: D-cache enable
9	ICEN	I-cache enable 0: I-cache disable 1: I-cache enable
8:5	Reserved	Must be kept at reset value.
4	PFEN	Pre-fetch enable 0: Pre-fetch disable 1: Pre-fetch enable
3	Reserved	Must be kept at reset value.
2:0	WSCNT[2:0]	Wait state counter register These bits is set and reset by software. 000: 0 wait state added

001: 1 wait state added
010: 2 wait state added
011: 3 wait state added
100: 4 wait state added
101 ~111: reserved

2.4.2. Unlock key register (FMC_KEY)

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_CTL unlock register These bits are only be written by software. Write KEY[31:0] with keys to unlock FMC_CTL 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_CTL option bytes operation unlock register These bits are only be written by software. Write OBKEY[31:0] with keys to unlock option bytes command in the FMC_CTL register.

2.4.4. Status register (FMC_STAT)

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	PGAERR	PGERR	Reserved	BUSY
										rc_w1	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	PGAERR	Program alignment error flag bit This bit is set by hardware when write data is not alignment. The software can clear it by writing 1.
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.

2.4.5. Control register (FMC_CTL)

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 the FMC_OBKEY register. This bit can be cleared by software.
8	Reserved	Must be kept at reset value.
7	LK	FMC_CTL lock bit This bit is cleared by hardware when right sequence written to the FMC_KEY 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 command bit This bit is set or cleared by software 0: no effect

		1: main flash mass erase command
1	PER	Main flash page erase command bit This bit is set or clear by software 0: no effect 1: main flash page erase command
0	PG	Main flash program command bit This bit is set or clear by software 0: no effect 1: main flash program command

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

2.4.6. Address register (FMC_ADDR)

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 to be erased/programmed.

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[15:0] 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[31:0] of option bytes block after system reset

2.4.9. 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	Field	Descriptions

31:0	PID[31:0]	Product reserved ID code register 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. Backup registers (BKP)

3.1. 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.

3.2. 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.

3.3. Function overview

3.3.1. RTC clock calibration

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

3.3.2. 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.

3.4. Register definition

BKP base address: 0x4000 6C00

3.4.1. 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.

3.4.2. 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
CALDIR	CCOSEL	Reserved				ROSEL	ASOEN	COEN	RCCV[6:0]						
rw	rw					rw	rw	rw	rw						

Bits	Fields	Descriptions
15	CALDIR	RTC clock calibration direction 0: Slowed down 1: Speed up This bit is reset only by a Backup domain reset.
14	CCOSEL	RTC clock output selection 0: RTC clock div 64 1: RTC clock This bit is reset only by a POR.
13:10	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 the RTC clock or RTC clock divided by 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.
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. This bit is reset only by a Backup domain reset.

3.4.3. 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.

3.4.4. 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.

4. Power management unit (PMU)

4.1. Overview

The power consumption is regarded as one of the most important issues for the devices of GD32E50x series. According to the Power management unit (PMU), provides five types of power saving modes, including Sleep, Deep-sleep, Deep-sleep 1, Deep-sleep 2 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 GD32E50x devices, there are three power domains, including V_{DD} / V_{DDA} domain, 1.1V domain, and Backup domain, as is shown in ***Figure 4-1. Power supply overview***. 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 1.1V 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.

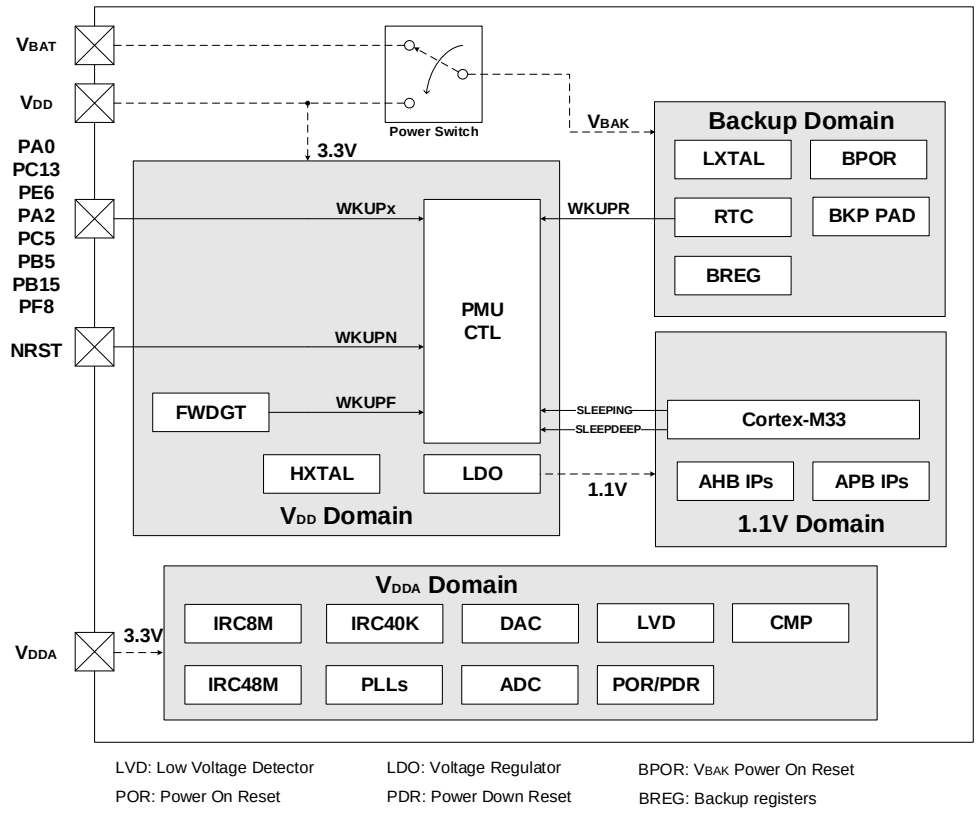
4.2. Characteristics

- Three power domains: V_{BAK}, V_{DD} / V_{DDA} and 1.1V power domains.
- Five power saving modes: Sleep, Deep-sleep, Deep-sleep 1, Deep-sleep 2 and Standby modes.
- Internal Voltage regulator (LDO) supplies around 1.1V voltage source for 1.1V 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.
- Ultra power saving for low-driver mode in Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode. And high-driver mode for high frequency.

4.3. Function overview

Figure 4-1. Power supply overview provides details on the internal configuration of the PMU and the relevant power domains.

Figure 4-1. Power supply overview



4.3.1. 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®-M33 can setup the RTC register with an expected alarm time and enable the alarm function and according EXTI lines to achieve the RTC alarm event. After entering the power saving mode for a certain amount of

time, the RTC alarm 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 ***Real-time clock (RTC)***.
- 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 ***Real-time clock (RTC)***.
- 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).

4.3.2. V_{DD} / V_{DDA} 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), FWDGT (free watchdog timer), all pads except PC13 / PC14 / PC15, etc. V_{DDA} domain includes POR / PDR (power on / down reset), ADC / DAC (AD / DA Converter), CMP (Comparator), IRC8M (internal 8MHz RC oscillator), IRC48M (internal 48MHz RC oscillator), IRC40K (internal 40KHz RC oscillator), PLLs (phase locking loop), LVD (low voltage detector), etc.

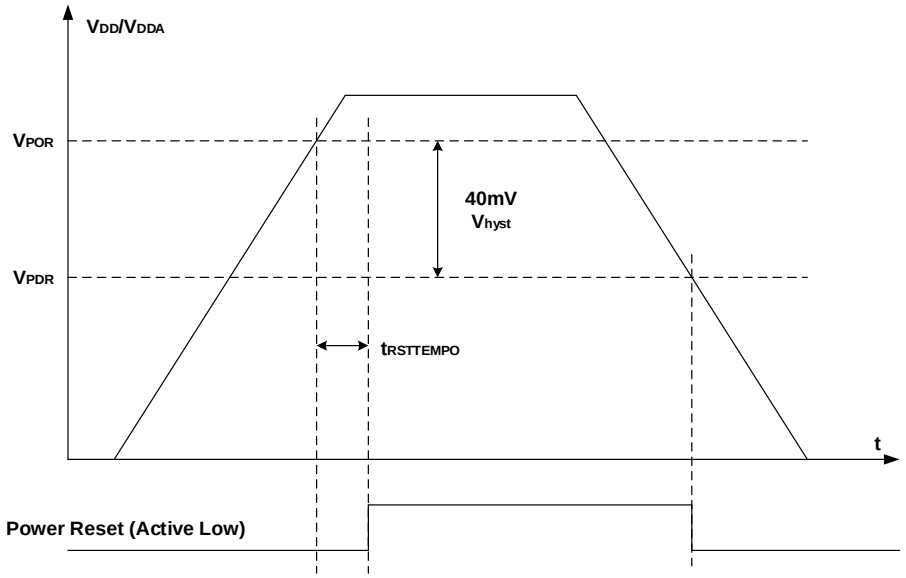
V_{DD} domain

The LDO, which is implemented to supply power for the 1.1V 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 / Deep-sleep 1 / Deep-sleep 2 mode (on or low power), and in the Standby mode (power off).

V_{DDA} domain

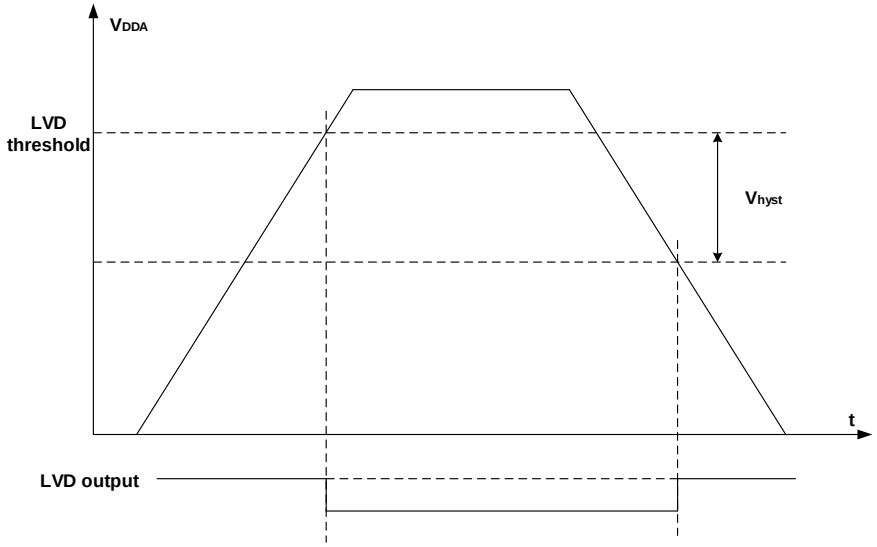
The POR / PDR circuit is implemented to detect V_{DDA} and generate the power reset signal which resets the whole chip except the Backup domain when the supply voltage is lower than the specified threshold. ***Figure 4-2. Waveform of the POR / PDR*** shows the relationship between the supply voltage and the power reset signal. V_{POR}, indicates the threshold of power on reset, while V_{PDR}, means the threshold of power down reset. Refer to the datasheet for the hysteresis voltage (V_{hyst}) value.

Figure 4-2. Waveform of the POR / PDR



The LVD is used to detect whether the V_{DDA} supply voltage is lower than a programmed threshold selected by the LVDT[2:0] bits in the Power control register(PMU_CTL0). The LVD is enabled by setting the LVDEN bit, and LVDF bit, which in the Power status register(PMU_CS0), indicates if 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 4-3. 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. Refer to the datasheet for the hysteresis voltage (V_{hyst}) value.

Figure 4-3. 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, when the V_{DD} and V_{DDA} are provided by different power supplies, the difference between V_{DD} and V_{DDA} during power-up and running time should not exceed 0.3V.

To ensure a high accuracy on ADC and DAC, the ADC / DAC independent external reference voltage should be connected to V_{REF+} / V_{REF-} pins. According to the different packages, V_{REF+} pin can be connected to V_{DDA} pin, or external reference voltage which refers to ***Table 13-2. ADC input pins definition*** and ***Table 14-1. DAC I/O description*** V_{REF-} pin must be connected to V_{SSA} pin. The V_{REF+} pin is only available on no less than 100-pin packages, or else the V_{REF+} pin is not available and internally connected to V_{DDA}. The V_{REF-} pin is only available on no less than 100-pin packages, or else the V_{REF-} pin is not available and internally connected to V_{SSA}.

4.3.3. 1.1V power domain

The main functions that include Cortex®-M33 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 1.1V is powered up, the POR will generate a reset sequence on the 1.1V 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.

High-driver mode

If the 1.1V power domain runs with high frequency and opens many functions, it is recommended to enter high-driver mode. The following steps are needed when using high-driver mode.

- IRC8M or HXTAL selected as system clock.
- Set HDEN bit in PMU_CTL0 register to 1 to open high-driver mode.
- Wait HDRF bit be set to 1 in PMU_CS0 register.
- Set HDS bit in PMU_CTL0 register to 1 to switch LDO to high-driver mode.
- Wait HDSRF bit be set to 1 in PMU_CS0 register. And enter high-driver mode.
- Running the application at high frequency.

The high-driver mode exit by resetting HDEN and HDS bits in PMU_CTL0 register after IRC8M or HXTAL selected as system clock. The high-driver mode exit automaticly when exiting from Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode.

4.3.4. Power saving modes

After a system reset or a power reset, the GD32E50x 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, and PCLK2) or gating the clocks of the unused peripherals. Besides, five power saving modes are provided to achieve even lower power consumption, they are Sleep mode, Deep-sleep mode, Deep-sleep 1 mode, Deep-sleep 2 mode and Standby mode.

Sleep mode

The Sleep mode is corresponding to the SLEEPING mode of the Cortex®-M33. In Sleep mode, only clock of Cortex®-M33 is off. To enter the Sleep mode, it is only necessary to clear the SLEEPDEEP bit in the Cortex®-M33 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®-M33 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®-M33 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®-M33. In Deep-sleep mode, all clocks in the 1.1V domain are off, and all of IRC8M, IRC48M, 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_CTL0 register. Before entering the Deep-sleep mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M33 System Control Register, and clear the STBMOD bit in the PMU_CTL0 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®-M33 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.

The low-driver mode in Deep-sleep mode can be entered by configuring the LDEN, LDNP, LDLP, LDOLP bits in the PMU_CTL0 register. The low-driver mode provides lower drive capability, and the low-power mode take lower power.

Normal-driver / Normal-power: The Deep-sleep mode is not in low-driver mode by configure LDEN to 00 in the PMU_CTL0 register, and not in low-power mode depending on the LDOLP

bit reset in the PMU_CTL0 register.

Normal-driver / Low-power: The Deep-sleep mode is not in low-driver mode by configure LDEN to 00 in the PMU_CTL0 register. The low-power mode enters depending on the LDOLP bit set in the PMU_CTL0 register.

Low-driver / Normal-power: The low-driver mode in Deep-sleep mode when the LDO in normal-power mode depending on the LDOLP bit reset in the PMU_CTL0 register enters by configure LDEN to 0b11 and LDNP to 1 in the PMU_CTL0 register.

Low-driver / Low-power: The low-driver mode in Deep-sleep mode when the LDO in low-power mode depending on the LDOLP bit set in the PMU_CTL0 register enters by configure LDEN to 0b11 and LDLP to 1 in the PMU_CTL0 register.

No Low-driver: The Deep-sleep mode is not in low-driver mode by configure LDEN to 00 in the PMU_CTL0 register.

Note: In order to enter Deep-sleep mode smoothly, all EXTI line pending status (in the EXTI_PD register) and related peripheral flags must be reset, refer to ***Table 7-3. EXTI source***. If not, the program will skip the entry process of Deep-sleep mode to continue to execute the following procedure.

Deep-sleep 1 mode

The Deep-sleep 1 mode is based on the SLEEPDEEP mode of the Cortex®-M33. In Deep-sleep 1 mode, all clocks in the 1.1V domain are off, and all of IRC8M, IRC48M, HXTAL and PLLs are disabled. The power of COREOFF1 domain is cut off. The contents of registers in COREOFF1 domain are lost. The LDO can operate normally or in low power mode depending on the LDOLP bit in the PMU_CTL0 register. Before entering the Deep-sleep 1 mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M33 System Control Register, clear the STBMOD bit in the PMU_CTL0 register, and set DPMOD1 bit in the PMU_CTL1 register. Then, the device enters the Deep-sleep 1 mode after a WFI or WFE instruction is executed. If the Deep-sleep 1 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®-M33 Technical Reference Manual). When exiting the Deep-sleep 1 mode, the IRC8M is selected as the system clock. Waking up from Deep-sleep 1 mode needs an additional delay to power on COREOFF1 domain. Notice that an additional wakeup delay will be incurred if the LDO operates in low power mode.

The low-driver mode in Deep-sleep 1 mode can be entered by configuring the LDEN, LDNP, LDLP, LDOLP bits in the PMU_CTL0 register. The low-driver mode provides lower drive capability, and the low-power mode take lower power.

Note:

1. COREOFF1 domain includes: ENET, SHRTIMER, USBHS, TMU modules.
2. COREOFF0 domain includes: 1.1V power domain modules except FMC, PMU, RCU, EXTI, GPIO, DBG, FWDGT, WWDGT, USART5, I2C2, and modules in COREOFF1

domain; includes SRAM (except the first 32K).

Deep-sleep 2 mode

The Deep-sleep 2 mode is based on the SLEEPDEEP mode of the Cortex®-M33. In Deep-sleep 2 mode, all clocks in the 1.1V domain are off, and all of IRC8M, IRC48M, HXTAL and PLLs are disabled. The power of COREOFF0/COREOFF1 domain is cut off. The contents of SRAM except for the first 32K and registers in COREOFF0/COREOFF1 domain are lost. The LDO can operate normally or in low power mode depending on the LDOLP bit in the PMU_CTL0 register. Before entering the Deep-sleep 2 mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M33 System Control Register, clear the STBMOD bit in the PMU_CTL0 register, and set DPMOD2 bit in the PMU_CTL1 register. Then, the device enters the Deep-sleep 2 mode after a WFI or WFE instruction is executed. If the Deep-sleep 2 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®-M33 Technical Reference Manual). Waking up from Deep-sleep 2 mode needs an additional delay to power on COREOFF0/COREOFF1 domain. Notice that an additional wakeup delay will be incurred if the LDO operates in low power mode.

The low-driver mode in Deep-sleep 2 mode can be entered by configuring the LDEN, LDNP, LDLP, LDOLP bits in the PMU_CTL0 register. The low-driver mode provides lower drive capability, and the low-power mode take lower power.

Note: When exiting from the Deep-sleep 2 mode, the Cortex®-M33 will execute instruction code from the 0x00000000 address.

Standby mode

The Standby mode is based on the SLEEPDEEP mode of the Cortex®-M33, too. In Standby mode, the whole 1.1V domain is power off, the LDO is shut down, and all of IRC8M, IRC48M, HXTAL and PLL are disabled. Before entering the Standby mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M33 System Control Register, and set the STBMOD bit in the PMU_CTL0 register, and clear WUF bit in the PMU_CS0 register. Then, the device enters the Standby mode after a WFI or WFE instruction is executed, and the STBF status flag in the PMU_CS0 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 pins. The Standby mode achieves the lowest power consumption, but spends longest time to wake up. Besides, the contents of SRAM and registers in 1.1V power domain are lost in Standby mode. When exiting from the Standby mode, a power-on reset occurs and the Cortex®-M33 will execute instruction code from the 0x00000000 address.

Table 4-1. Power saving mode summary

Mode	Sleep	Deep-sleep	Deep-sleep 1	Deep-sleep 2	Standby
Descriptio	Only CPU clock	1. All clocks in the	1. All clocks in the	1. All clocks in the 1.1V	1. The 1.1V

Mode	Sleep	Deep-sleep	Deep-sleep 1	Deep-sleep 2	Standby
n	is off	1.1V domain are off. 2. Disable IRC8M, IRC48M, HXTAL and PLLs.	1.1V domain are off. 2. Disable IRC8M, IRC48M, HXTAL and PLLs. 3. SHRTIMER / USBHS / TMU / ENET power off.	domain are off. 2. Disable IRC8M, IRC48M, HXTAL and PLLs. 3. CPU / SRAM except for the first 32K / peripherals in COREOFF0 and COREOFF1 power off.	domain is power off. 2. Disable IRC8M, IRC48M, HXTAL and PLLs.
LDO Status	On (normal power mode)	On (normal or low power mode, normal or low driver mode)	On (normal or low power mode, normal or low driver mode)	On (normal or low power mode, normal or low driver mode)	Off
Configuration	SLEEPDEEP = 0	SLEEPDEEP = 1 STBMOD = 0	SLEEPDEEP = 1 STBMOD = 0 DPMOD1 = 1	SLEEPDEEP = 1 STBMOD = 0 DPMOD2 = 1	SLEEPDEEP = 1 STBMOD = 1, WURST=1
Entry	WFI or WFE	WFI or WFE	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	Any interrupt from EXTI lines for WFI Any event(or interrupt when SEVONPEND is 1) from EXTI 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 pins 3. FWDGT reset 4. RTC
Wakeup Latency	None	IRC8M wakeup time, LDO wakeup time added if LDO is in low power mode	IRC8M wakeup time, COREOFF1 power on time added, and LDO wakeup time added if LDO is in low power mode	IRC8M wakeup time, COREOFF0/COREOFF1 power on time added, and LDO wakeup time added if LDO is in low power mode	Power on sequence

Note: In Standby mode, all I/Os are in high-impedance state except NRST pin, PC13 pin when configured for RTC function, PC14 and PC15 pins when used as LXTAL crystal oscillator pins, and WKUP pins if enabled.

4.4. Register definition

PMU base address: 0x4000 7000

4.4.1. Control register 0 (PMU_CTL0)

Address offset: 0x00
Reset value: 0x0000 C000 (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												LDEN[1:0]		HDS	HDEN
												rw		rw	rw

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

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:18	LDEN[1:0]	Low-driver mode enable in Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode 00: Low-driver mode disable in Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode 01: Reserved 10: Reserved 11: Low-driver mode enable in Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode
17	HDS	High-driver mode switch Set this bit by software only when HDRF flag is set and IRC8M or HXTAL used as system clock. After this bit is set, the system enters High-driver mode. This bit can be cleared by software. And cleared by hardware when exit from Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode or when the HDEN bit is clear. 0: No High-driver mode switch 1: High-driver mode switch
16	HDEN	High-driver mode enable This bit is set by software only when IRC8M or HXTAL used as system clock. This bit is cleared by software or by hardware when exit from Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode. 0: High-driver mode disable 1: High-driver mode enable
15:12	Reserved	Must be kept at reset value.
11	LDNP	Low-driver mode when use normal power LDO 0: normal driver when use normal power LDO

		1: Low-driver mode enabled when LDEN is 11 and use normal power LDO
10	LDLP	Low-driver mode when use low power LDO. 0: normal driver when use low power LDO 1: Low-driver mode enabled when LDEN is 11 and use low power LDO
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: 2.1V 001: 2.3V 010: 2.4V 011: 2.6V 100: 2.7V 101: 2.9V 110: 3.0V 111: 3.1V
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 / Deep-sleep 1 / Deep-sleep 2 mode when the Cortex®-M33 enters SLEEPDEEP mode 1: Enter the Standby mode when the Cortex®-M33 enters SLEEPDEEP mode
0	LDOLP	LDO Low Power Mode 0: The LDO operates normally during the Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode 1: The LDO is in low power mode during the Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode

Note: Some peripherals may work with the IRC8M clock in the Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode. In this case, the LDO automatically switches from the low power mode to the normal mode and remains in this mode until the peripheral stop working.

4.4.2. Control and status register 0 (PMU_CS0)

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												LDRF[1:0]		HDSRF	HDRF
												rc_w1		r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUPEN7	Reserved	WUPEN5	WUPEN4	WUPEN3	WUPEN2	WUPEN1	WUPEN0	WUPEN6	Reserved				LVDF	STBF	WUF
rw		rw	rw	rw	rw	rw	rw	rw					r	r	r

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:18	LDRF[1:0]	Low-driver mode ready flag These bits are set by hardware when enter Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode and the LDO in Low-driver mode. These bits are cleared by software when write 11. 00: normal driver in Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode 01: Reserved 10: Reserved 11: Low-driver mode in Deep-sleep / Deep-sleep 1 / Deep-sleep 2 mode
17	HDSRF	High-driver switch ready flag 0: High-driver switch not ready 1: High-driver switch ready
16	HDRF	High-driver ready flag 0: High-driver not ready 1: High-driver ready
15	WUPEN7	WKUP Pin7(PF8) Enable 0: Disable WKUP pin7 function 1: Enable WKUP pin7 function If WUPEN7 is set before entering the power saving mode, a rising edge on the WKUP pin7 wakes up the system from the power saving mode. As the WKUP pin7 is active high, the WKUP pin7 is internally configured to input pull down mode. And set this bit will trigger a wakup event when the input is aready high.

14	Reserved	Must be kept at reset value.
13	WUPEN5	WKUP Pin5(PB5) Enable 0: Disable WKUP pin5 function 1: Enable WKUP pin5 function If WUPEN5 is set before entering the power saving mode, a rising edge on the WKUP pin5 wakes up the system from the power saving mode. As the WKUP pin5 is active high, the WKUP pin5 is internally configured to input pull down mode. And set this bit will trigger a wakup event when the input is aready high.
12	WUPEN4	WKUP Pin4(PC5) Enable 0: Disable WKUP pin4 function 1: Enable WKUP pin4 function If WUPEN4 is set before entering the power saving mode, a rising edge on the WKUP pin4 wakes up the system from the power saving mode. As the WKUP pin4 is active high, the WKUP pin4 is internally configured to input pull down mode. And set this bit will trigger a wakup event when the input is aready high.
11	WUPEN3	WKUP Pin3(PA2) Enable 0: Disable WKUP pin3 function 1: Enable WKUP pin3 function If WUPEN3 is set before entering the power saving mode, a rising edge on the WKUP pin3 wakes up the system from the power saving mode. As the WKUP pin3 is active high, the WKUP pin3 is internally configured to input pull down mode. And set this bit will trigger a wakup event when the input is aready high.
10	WUPEN2	WKUP Pin2(PE6) Enable 0: Disable WKUP pin2 function 1: Enable WKUP pin2 function If WUPEN2 is set before entering the power saving mode, a rising edge on the WKUP pin2 wakes up the system from the power saving mode. As the WKUP pin2 is active high, the WKUP pin2 is internally configured to input pull down mode. And set this bit will trigger a wakup event when the input is aready high.
9	WUPEN1	WKUP Pin1(PC13) Enable 0: Disable WKUP pin1 function 1: Enable WKUP pin1 function If WUPEN1 is set before entering the power saving mode, a rising edge on the WKUP pin1 wakes up the system from the power saving mode. As the WKUP pin1 is active high, the WKUP pin1 is internally configured to input pull down mode. And set this bit will trigger a wakup event when the input is aready high.
8	WUPEN0	WKUP Pin0(PA0) Enable 0: Disable WKUP pin0 function 1: Enable WKUP pin0 function If WUPEN0 is set before entering the power saving mode, a rising edge on the WKUP pin0 wakes up the system from the power saving mode. As the WKUP pin0 is active high, the WKUP pin0 is internally configured to input pull down mode.

		And set this bit will trigger a wakup event when the input is aready high.
7	WUPEN6	WKUP Pin6(PB15) Enable 0: Disable WKUP pin6 function 1: Enable WKUP pin6 function If WUPEN6 is set before entering the power saving mode, a rising edge on the WKUP pin6 wakes up the system from the power saving mode. As the WKUP pin6 is active high, the WKUP pin6 is internally configured to input pull down mode. And set this bit will trigger a wakup event when the input is aready high.
6:3	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_CTL0 register.
0	WUF	Wakeup Flag 0: No wakeup event has been received 1: Wakeup event occurred from the WKUP pins or the RTC alarm event This bit is cleared only by a POR / PDR or by setting the WURST bit in the PMU_CTL0 register.

4.4.3. Control register 1 (PMU_CTL1)

Address offset: 0x08

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														DPMOD2	DPMOD1
														rw	rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.

1	DPMOD2	Deep-sleep 2 mode enable 0: Not care 1:Go to Deep-sleep 2 mode when SLEEPDEEP bit is set and the STBMOD bit is clear
0	DPMOD1	Deep-sleep 1 mode enable 0: Not care 1:Go to Deep-sleep 1 mode when the SLEEPDEEP bit is set and the STBMOD bit is clear and the DPMOD2 bit is clear

4.4.4. Control and status register 1 (PMU_CS1)

Address offset: 0x0C
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														DPF2	DPF1
														rc_w0	rc_w0

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	DPF2	Deep-sleep 2 mode status flag. This bit set by hardware when enter Deep-sleep 2 mode. And clear by software when write 0.
0	DPF1	Deep-sleep1 mode status flag. This bit set by hardware when enter Deep-sleep 1 mode. And clear by software when write 0.

5. Reset and clock unit (RCU)

High density reset and clock control unit (RCU)

5.1. Reset control unit (RCTL)

5.1.1. Overview

GD32E50x 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. These 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 1.1V power. The reset service routine vector is fixed at address 0x0000_0004 in the memory map.

System reset

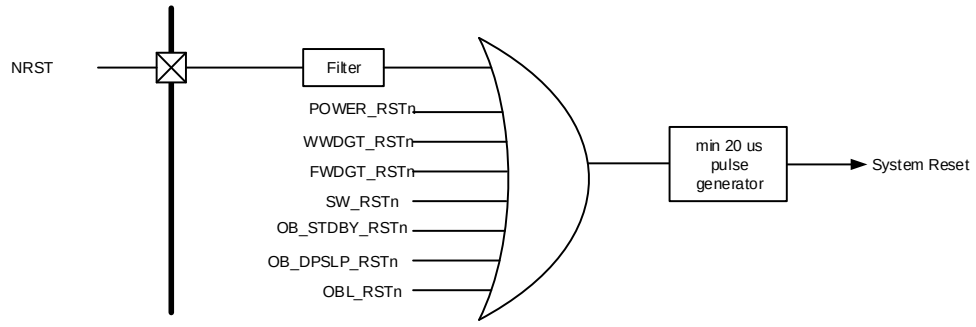
A system reset is generated by the following events:

- 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®-M33 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).

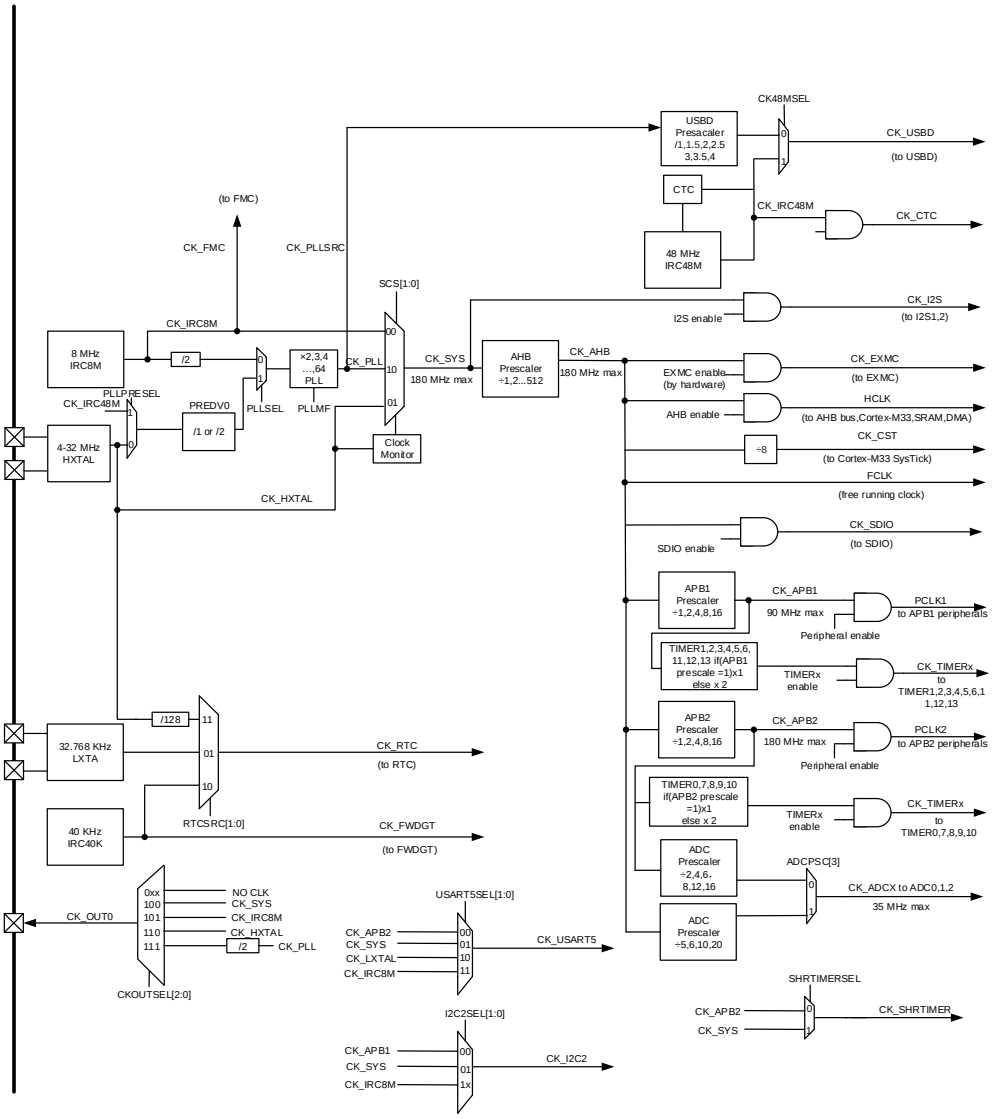
5.2. Clock control unit (CCTL)

5.2.1. Overview

The clock control unit provides a range of frequencies and clock functions. These include an Internal 8M RC oscillator (IRC8M), an Internal 48M RC oscillator (IRC48M), 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®-M33 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 180 MHz.

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 180 MHz / 180 MHz / 90 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 or by the clock of AHB divided by 5, 6, 10, 20, which defined by ADCPSC in RCU_CFG0 and RCU_CFG1 register.

The USART5 is clocked by IRC8M clock or LXTAL clock or system clock or APB2 clock, which selected by USART5SEL bits in configuration register 2 (RCU_CFG2).

The I2C2 is clocked by IRC8M clock or system clock or APB1 clock, which selected by I2C2SEL bits in configuration register 2 (RCU_CFG2).

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 CK48M. The CK48M is selected from the clock of CK_PLL or the clock of IRC48M by CK48MSEL bit in RCU_ADDCTL register.

The CTC is clocked by the clock of IRC48M. The IRC48M can be automatically trimmed by CTC unit.

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 1.1V 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.

The SHRTIMER is clocked by the clock of CK_APB2 or CK_SYS which defined by SHRTIMERSEL bit in RCU_CFG1 register.

If the SHRTIMER high-resolution mode is not required, the SHRTIMERSEL bit in the RCU_CFG1 register can remain cleared. In this case, the CNTCKDIV[2:0] value in the SHRTIMER_MTCTL0 register must be at least 5 (prescaler ratio of 64 or greater).

When high-resolution mode is required for the SHRTIMER, the source of system clock source must be selected as PLL, and the SHRTIMERSEL bit in the RCU_CFG1 register must be set to 1 to select CK_SYS as the clock source. In this scenario, any value can be used for CNTCKDIV[2:0] in the SHRTIMER_MTCTL0 register.

Note: For high-resolution configuration, the AHB and APB2 prescalers (AHBPSC[3:0] and APB2PSC[2:0] bits in the RCU_CFG0 register) must be set to maintain a proportional relationship of 1, 2, or 4 between the system clock CK_SYS and the APB2 clock PCLK2.

5.2.2. Characteristics

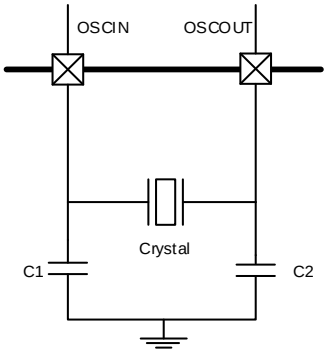
- 4 to 32 MHz High Speed crystal oscillator (HXTAL).
- Internal 8 MHz RC oscillator (IRC8M).
- Internal 48 MHz RC oscillator (IRC48M).
- 32,768 Hz Low Speed crystal oscillator (LXTAL).
- Internal 40KHz RC oscillator (IRC40K).
- PLL clock source can be HXTAL, IRC8M, IRC48M.
- HXTAL clock monitor.

5.2.3. Function overview

High speed crystal oscillator (HXTAL)

The high speed external crystal oscillator (HXTAL), which has a frequency from 4 to 32 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.

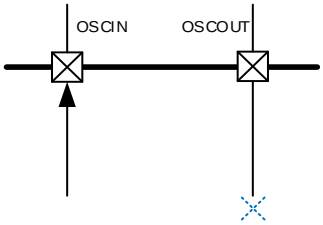
Figure 5-3. HXTAL clock source



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. During bypass mode, the signal is connected to OSCIN, and OSCOUT remains in the suspended state, as shown in ***Figure 5-4. HXTAL clock source in bypass mode***. The CK_HXTAL is equal to the external clock which drives the OSCIN pin.

Figure 5-4. HXTAL clock source in bypass mode



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.

Internal 48M RC oscillators (IRC48M)

The internal 48M RC oscillator, IRC48M, has a fixed frequency of 48 MHz. The IRC48M oscillator provides a lower cost type clock source as no external components are required when USBD used. The IRC48M RC oscillator can be switched on or off using the IRC48MEN bit in the RCU_ADDCTL register. The IRC48MSTB flag in the RCU_ADDCTL register is used to indicate if the internal 48M RC oscillator is stable. An interrupt can be generated if the related interrupt enable bit, IRC48MSTBIE, in the RCU_ADDINT register, is set when the IRC48M becomes stable. The IRC48M clock is used for the clocks of USBD.

The frequency accuracy of the IRC48M can be calibrated by the manufacturer, but its operating frequency is still not enough accurate because the USB need the frequency must between 48MHz with 500ppm accuracy. A hardware automatically dynamic trim performed in CTC unit adjust the IRC48M to the needed frequency.

Phase locked loop (PLL)

There is one internal Phase Locked Loop, the 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 rree 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®-M33. 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 180 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 1.1V domain voltage in Deep-sleep mode can be controlled by DSLPVS[2:0] bit in the Deep-sleep mode voltage register (RCU_DSV).

Table 5-2. 1.1V domain voltage selected in deep-sleep mode

DSLPVS[2:0]	Deep-sleep mode voltage(V)
000	1.0
001	0.9
010	0.8
011	0.7

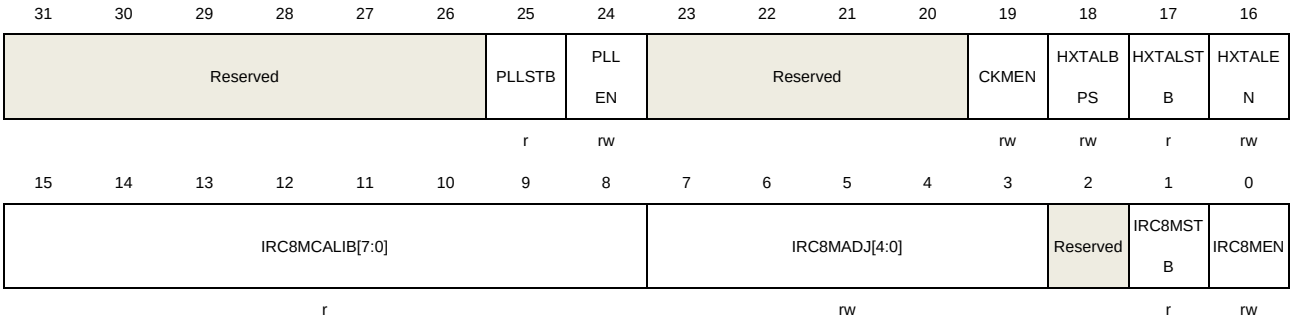
5.3. Register definition

RCU base address: 0x4002 1000

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).



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 ~ 32 MHz crystal oscillator (HXTAL) clock monitor 1: Enable the High speed 4 ~ 32 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 ~ 32 MHz crystal oscillator disabled 1: High speed 4 ~ 32 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

USBDPSC[2]	PLLMF[5]	Reserved	ADCPSC[2]	PLLMF[4]	CKOUT0SEL[2:0]	USBDPSC[1:0]	PLLMF[3:0]	PREDV0	PLLSEL
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

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	USBDPSC[2]	Bit 2 of USBDPSC see bits 23:22 of RCU_CFG0
30	PLLMF[5]	Bit 5 of PLLMF see bits 21:18 of RCU_CFG0
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: Internal 8MHz RC 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. 000: CK_USBD = CK_PLL / 1.5 001: CK_USBD = CK_PLL 010: CK_USBD = CK_PLL / 2.5 011: CK_USBD = CK_PLL / 2 100: CK_USBD = CK_PLL / 3 101: CK_USBD = CK_PLL / 3.5 11x: CK_USBD = CK_PLL / 4
21:18	PLLMF[3:0]	The PLL clock multiplication factor Bit 27, bit 30 of RCU_CFG0 and these bits are written by software to define the PLL multiplication factor. **Note**: The PLL output frequency must not exceed 180 MHz. 000000: CK_SYS = CK_PLL x 2 000001: CK_SYS = CK_PLL x 3

000010: CK_SYS = CK_PLL x 4
000011: CK_SYS = CK_PLL x 5
000100: CK_SYS = CK_PLL x 6
000101: CK_SYS = CK_PLL x 7
000110: CK_SYS = CK_PLL x 8
000111: CK_SYS = CK_PLL x 9
001000: CK_SYS = CK_PLL x 10
001001: CK_SYS = CK_PLL x 11
001010: CK_SYS = CK_PLL x 12
001011: CK_SYS = CK_PLL x 13
001100: CK_SYS = CK_PLL x 14
001101: CK_SYS = CK_PLL x 15
001110: CK_SYS = CK_PLL x 16
001111: CK_SYS = CK_PLL x 16
010000: CK_SYS = CK_PLL x 17
010001: CK_SYS = CK_PLL x 18
010010: CK_SYS = CK_PLL x 19
010011: CK_SYS = CK_PLL x 20
010100: CK_SYS = CK_PLL x 21
010101: CK_SYS = CK_PLL x 22
010110: CK_SYS = CK_PLL x 23
010111: CK_SYS = CK_PLL x 24
011000: CK_SYS = CK_PLL x 25
011001: CK_SYS = CK_PLL x 26
011010: CK_SYS = CK_PLL x 27
011011: CK_SYS = CK_PLL x 28
011100: CK_SYS = CK_PLL x 29
011101: CK_SYS = CK_PLL x 30
011110: CK_SYS = CK_PLL x 31
011111: CK_SYS = CK_PLL x 32
100000: CK_SYS = CK_PLL x 33
100001: CK_SYS = CK_PLL x 34
…
111110: CK_SYS = CK_PLL x 63
111111: CK_SYS = CK_PLL x 64

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 or IRC48M(PLLPRESEL of RCU_CFG1 register) selected as source

		clock of PLL
15:14	ADCPSC[1:0]	ADC clock prescaler selection These bits, bit 28 of RCU_CFG0 and bit 29 of RCU_CFG1 are written by software to define the ADC prescaler factor.Set and cleared by software. 0000: (CK_APB2 / 2) selected 0001: (CK_APB2 / 4) selected 0010: (CK_APB2 / 6) selected 0011: (CK_APB2 / 8) selected 0100: (CK_APB2 / 2) selected 0101: (CK_APB2 / 12) selected 0110: (CK_APB2 / 8) selected 0111: (CK_APB2 / 16) selected 1x00: (CK_AHB / 5) selected 1x01: (CK_AHB / 6) selected 1x10: (CK_AHB / 10) selected 1x11: (CK_AHB / 20) 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. 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	IRC40K STBIC
								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 Set and reset by software to enable/disable LXTAL stabilization interrupt 0: Disable the LXTAL stabilization interrupt 1: Enable the LXTAL stabilization interrupt
8	IRC40KSTBIE	IRC40K stabilization interrupt enable Set and reset by software to enable/disable IRC40K stabilization interrupt 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 ~ 32 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		SHRTIMERST	USART5RST	Reserved						TIMER10RST	TIMER9RST	TIMER8RST	Reserved		
		rw	rw							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
31:30	Reserved	Must be kept at reset value.
29	SHRTIMERST	SHRTIMER reset This bit is set and reset by software. 0: No reset 1: Reset the SHRTIMER
28	USART5RST	USART5 reset This bit is set and reset by software. 0: No reset 1: Reset the USART5
27:22	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 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

5.3.5. 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	CAN1RST	CAN0RST	I2C2RST	USBDRST	I2C1RST	I2C0RST	UART4RST	UART3RST	USART2RST	USART1RST	Reserved
		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
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	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	I2C2RST	I2C2 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C2
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
SQPIEN	Reserved														
rw															
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	SQPIEN	SQPI clock enable This bit is set and reset by software. 0: Disabled SQPI clock 1: Enabled SQPI clock
30:11	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		SHRTIMEREN	USART5EN	Reserved						TIMER10EN	TIMER9EN	TIMER8EN	Reserved		
		rw	rw							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
31:30	Reserved	Must be kept at reset value
29	SHRTIMEREN	SHRTIMER clock enable This bit is set and reset by software. 0: Disabled SHRTIMER clock 1: Enabled SHRTIMER clock
28	USART5EN	USART5 clock enable This bit is set and reset by software. 0: Disabled USART5 clock 1: Enabled USART5 clock
27:22	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	CAN1EN	CAN0EN	I2C2EN	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	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	I2C2EN	I2C2 clock enable This bit is set and reset by software. 0: Disabled I2C2 clock 1: Enabled I2C2 clock
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 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 0 (PMU_CTL0) 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			LXTALDRI[1:0]		LXTALBPS	LXTALSTB	LXTALEN
rw						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
7:5	Reserved	Must be kept at reset value.
4:3	LXTALDRI[1:0]	LXTAL drive capability Set and reset by software. Backup domain reset resets this value. 00: Lower driving capability 01: Medium low driving capability 10: Medium high driving capability 11: Higher driving capability (reset value) **Note:** The LXTALDRI is not in bypass mode.
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

5.3.10. 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
LPRSTF	WWDGTRSTF	FWDGTRSTF	SWRSTF	PORRSTF	EPRSTF	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														IRC40KSTB	IRC40KEN
														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.

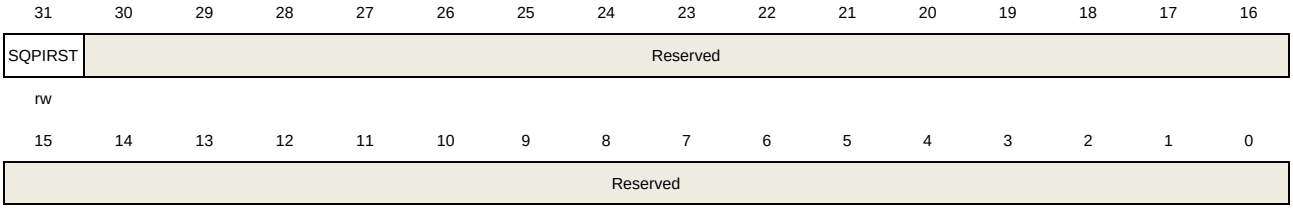
		0: No external pin reset generated 1: External pin reset generated
25	Reserved	Must be kept at reset value.
24	RSTFC	Reset flag clear This bit is set by software to clear all reset flags. 0: Not clear reset flags 1: Clear reset flags
23:2	Reserved	Must be kept at reset value.
1	IRC40KSTB	IRC40K stabilization 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. 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).



Bits	Fields	Descriptions
31	SQPIRST	SQPI reset This bit is set and reset by software. 0: No reset 1: Reset the SQPI
30:0	Reserved	Must be kept at reset value.

5.3.12. 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	PLLPRESEL	ADCPSC[3]	Reserved									SHRTIMERSEL	Reserved		
	rw	rw										rw			

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

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30	PLLPRESEL	PLL clock source preselection 0: HXTAL selected as PLL source clock 1: CK_IRC48M selected as PLL source clock
29	ADCPSC[3]	Bit 3 of ADCPSC see bits 15:14 of RCU_CFG0 and bit 28 of RCU_CFG0
28:20	Reserved	Must be kept at reset value.
19	SHRTIMERSEL	SHRTIMER Clock Source Selection Set and reset by software to control the SHRTIMER clock source. 0: APB2 clock selected as SHRTIMER source clock 1: System clock selected as SHRTIMER source clock
18:0	Reserved	Must be kept at reset value.

5.3.13. 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 1.0V in Deep-sleep mode

001: The core voltage is 0.9V in Deep-sleep mode

010: The core voltage is 0.8V in Deep-sleep mode

011: The core voltage is 0.7V in Deep-sleep mode

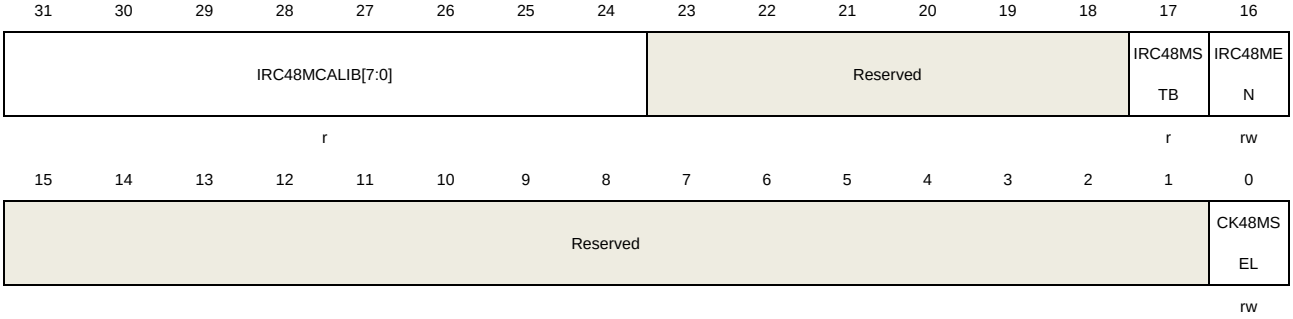
1xx: Reserved

5.3.14. Additional clock control register (RCU_ADDCTL)

Address offset: 0xC0

Reset value: 0x8000 0000

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



Bits	Fields	Descriptions
31:24	IRC48MCALIB [7:0]	Internal 48MHz RC oscillator calibration value register These bits are load automatically at power on.
23:18	Reserved	Must be kept at reset value.
17	IRC48MSTB	Internal 48MHz RC oscillator clock stabilization flag Set by hardware to indicate if the IRC48M oscillator is stable and ready for use. 0: IRC48M is not stable 1: IRC48M is stable
16	IRC48MEN	Internal 48MHz RC oscillator enable Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: IRC48M disable 1: IRC48M enable
15:2	Reserved	Must be kept at reset value.
0	CK48MSEL	48MHz clock selection Set and reset by software. This bit used to generate CK48M clock which select IRC48M clock or PLL48M clock. 0: Don't select IRC48M clock(use CK_PLL clock divided by USBDPSC) 1: Select IRC48M clock

5.3.15. Additional clock interrupt register (RCU_ADDINT)

Address offset: 0xCC

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									IRC48MSTBIC	Reserved					
									w						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	IRC48MSTBIE	Reserved							IRC48MSTBIF	Reserved					
	rw								r						

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value.
22	IRC48MSTBIC	Internal 48 MHz RC oscillator Stabilization interrupt clear Write 1 by software to reset the IRC48MSTBIF flag. 0: Not reset IRC48MSTBIF flag 1: Reset IRC48MSTBIF flag
21:15	Reserved	Must be kept at reset value.
14	IRC48MSTBIE	Internal 48 MHz RC oscillator stabilization interrupt enable Set and reset by software to enable/disable the IRC48M stabilization interrupt 0: Disable the IRC48M stabilization interrupt 1: Enable the IRC48M stabilization interrupt
13:7	Reserved	Must be kept at reset value.
6	IRC48MSTBIF	IRC48M stabilization interrupt flag Set by hardware when the Internal 48 MHz RC oscillator clock is stable and the IRC48MSTBIE bit is set. Reset by software when setting the IRC48MSTBIC bit. 0: No IRC48M stabilization interrupt generated 1: IRC48M stabilization interrupt generated
5:0	Reserved	Must be kept at reset value.

5.3.16. PLL clock spread spectrum control register (RCU_PLLSSCTL)

Address offset: 0xD0

Reset value: 0x0000 0000

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

The spread spectrum modulation is available only for the main PLL clock

The RCU_PLLSSCTL register must be written when the main PLL is disabled

This register is used to configure the PLL spread spectrum clock generation according to the following formulas:

MODCNT = round(f_{PLLIN}/4/f_{mod})

MODSTEP = round(mdamp*PLLN*2^{15}/(MODCNT*100))

Where f_{PLLIN} represents the PLL input clock frequency, f_{mod} represents the spread spectrum modulation frequency, mdamp represents the spread spectrum modulation amplitude expressed as a percentage, PLLN represents the PLL clock frequency multiplication factor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCGON	SS_TYPE	Reserved		MODSTEP[14:3]											
rw	rw			rw											

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

Bits	Fields	Descriptions
31	SSCGON	PLL spread spectrum modulation enable 0: Spread spectrum modulation disable 1: Spread spectrum modulation enable
30	SS_TYPE	PLL spread spectrum modulation type select 0: Center spread selected 1: Down spread selected
29:28	Reserved	Must be kept at reset value.
27:13	MODSTEP[14:0]	These bits configure PLL spread spectrum modulation profile amplitude and frequency. The following criteria must be met: MODSTEP*MODCNT≤2^{15}-1.
12:0	MODCNT[12:0]	These bits configure PLL spread spectrum modulation profile amplitude and frequency. The following criteria must be met: MODSTEP*MODCNT≤2^{15}-1.

5.3.17. Clock configuration register 2 (RCU_CFG2)

Address offset: 0xD4

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										I2C2SEL[1:0]		Reserved		USART5SEL[1:0]	
										rw				rw	

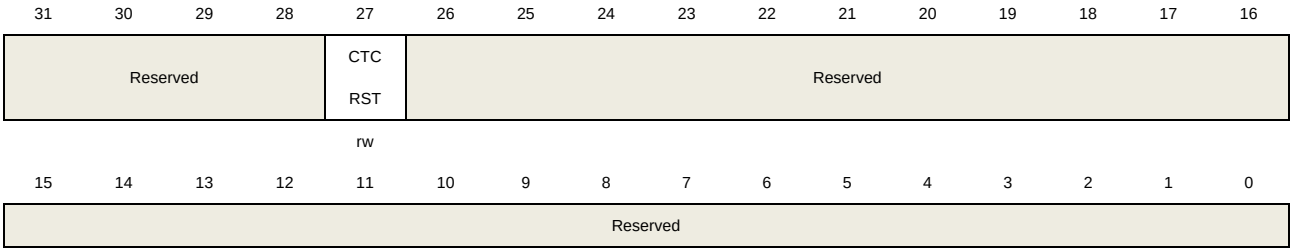
Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5:4	I2C2SEL[1:0]	I2C2 Clock Source Selection Set and reset by software to control the I2C2 clock source. 00: APB1 clock selected as I2C2 source clock 01: System clock selected as I2C2 source clock 1x: CK_IRC8M clock selected as I2C2 source clock
3:2	Reserved	Must be kept at reset value.
1:0	USART5SEL[1:0]	USART5 Clock Source Selection Set and reset by software to control the USART5 clock source. 00: CK_APB2 selected as USART5 source clock 01: CK_SYS selected as USART5 source clock 10: CK_LXTAL selected as USART5 source clock 11: CK_IRC8M selected as USART5 source clock

5.3.18. APB1 additional reset register (RCU_ADDAPB1RST)

Address offset: 0xE0

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27	CTCRST	CTC reset This bit is set and reset by software. 0: No reset 1: Reset CTC
26:0	Reserved	Must be kept at reset value.

5.3.19. APB1 additional enable register (RCU_ADDAPB1EN)

Address offset: 0xE4

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

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27	CTCEN	CTC clock enable This bit is set and reset by software. 0: Disabled CTC clock 1: Enabled CTC clock
26:0	Reserved	Must be kept at reset value.

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

5.4. Reset control unit (RCTL)

5.4.1. Overview

GD32E50x 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. These 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 1.1V 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:

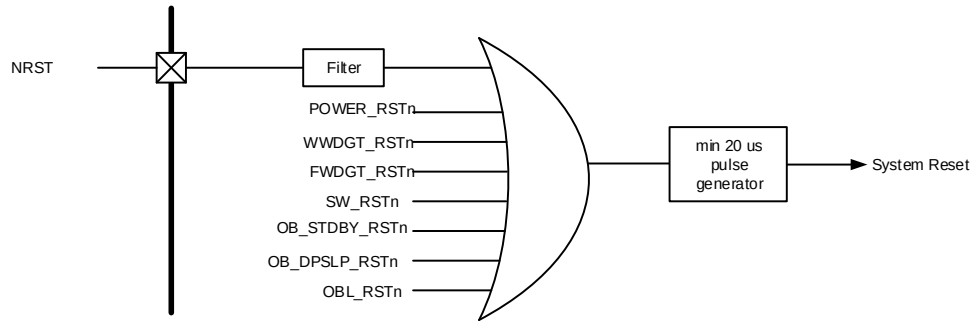
- A power reset (POWER_RSTn).
- A external pin reset (NRST).
- A window watchdog timer reset (WWDGT_RSTn).
- A free watchdog timer reset (FWDGT_RSTn).
- The SYSRESETREQ bit in Cortex®-M33 application interrupt and reset control register is set (SW_RSTn).
- Reset generated when entering Standby mode when 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 5-5. 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).

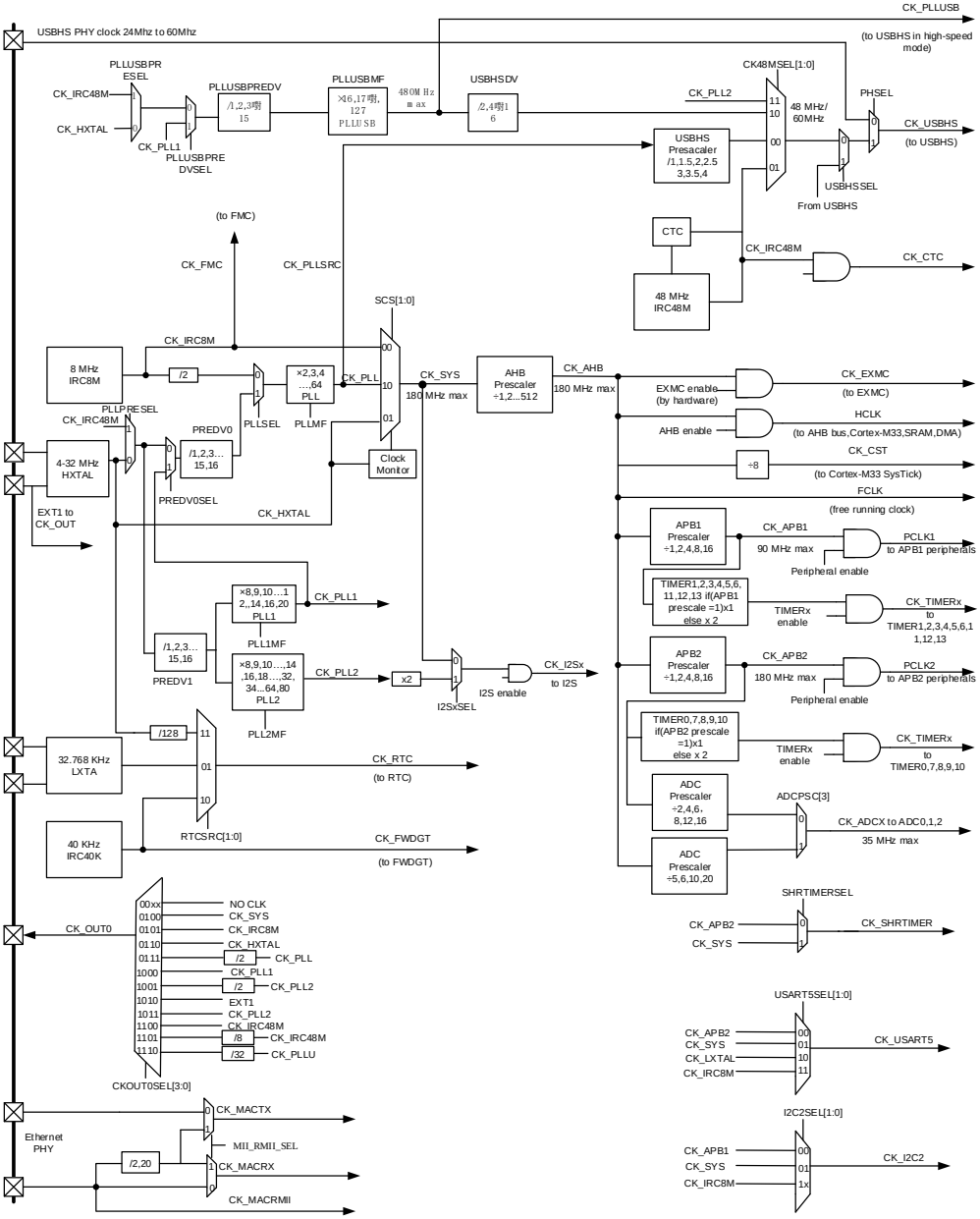
5.5. Clock control unit (CCTL)

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 Internal 48M RC oscillator (IRC48M), a High Speed crystal oscillator (HXTAL), a Low Speed Internal 40K RC oscillator (IRC40K), a Low Speed crystal oscillator (LXTAL), four Phase Lock Loop (PLL), a HXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AHB, APB and Cortex®-M33 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 180 MHz.

Figure 5-6. 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 180 MHz / 180 MHz / 90 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 or by the clock of AHB divided by 5, 6, 10, 20, which defined by ADCPSC in RCU_CFG0 and RCU_CFG1 register.

The USART5 is clocked by IRC8M clock or LXTAL clock or system clock or APB2 clock, which selected by USART5SEL bits in configuration register 2 (RCU_CFG2).

The I2C2 is clocked by IRC8M clock or system clock or APB1 clock, which selected by I2C2SEL bits in configuration register 2 (RCU_CFG2).

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 PLLUSB is clocked by the clock of HXTAL or the clock of IRC48M which defined by PLLUSBPRESEL bit in RCC_ADDCFG register. The maximum operating frequency of the CK_PLLUSB can be up to 480 MHz.

The USBHS is clocked by external PHY clock or internal crystal clock, which select by EMBPHY in USBHS_ GUSBCS register.

The CTC is clocked by the clock of IRC48M. The IRC48M can be automatically trimmed by CTC unit.

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 1.1V 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.

The SHRTIMER is clocked by the clock of CK_APB2 or CK_SYS which defined by SHRTIMERSEL bit in RCU_CFG1 register.

If the SHRTIMER high-resolution mode is not required, the SHRTIMERSEL bit in the RCU_CFG1 register can remain cleared. In this case, the CNTCKDIV[2:0] value in the SHRTIMER_MTCTL0 register must be at least 5 (prescaler ratio of 64 or greater).

When high-resolution mode is required for the SHRTIMER, the source of system clock source must be selected as PLL, and the SHRTIMERSEL bit in the RCU_CFG1 register must be set to 1 to select CK_SYS as the clock source. In this scenario, any value can be used for CNTCKDIV[2:0] in the SHRTIMER_MTCTL0 register.

Note: For high-resolution configuration, the AHB and APB2 prescalers (AHBPSC[3:0] and APB2PSC[2:0] bits in the RCU_CFG0 register) must be set to maintain a proportional relationship of 1, 2, or 4 between the system clock CK_SYS and the APB2 clock PCLK2.

5.5.2. Characteristics

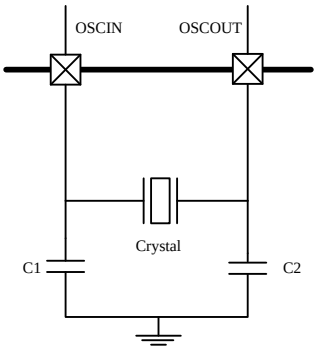
- 4 to 32 MHz high speed crystal oscillator (HXTAL).
- Internal 8 MHz RC oscillator (IRC8M).
- Internal 48 MHz RC oscillator (IRC48M).
- 32,768 Hz low speed crystal oscillator (LXTAL).
- Internal 40KHz RC oscillator (IRC40K).
- PLL clock source can be HXTAL, IRC8M orIRC48M.
- HXTAL clock monitor.

5.5.3. Function overview

High speed crystal oscillator (HXTAL)

The high speed external crystal oscillator (HXTAL), which has a frequency from 4 to 32 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.

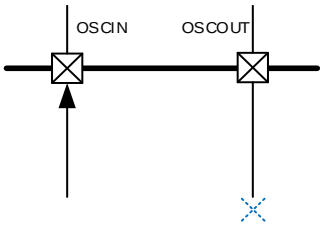
Figure 5-7. HXTAL clock source



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. During bypass mode, the signal is connected to OSCIN, and OSCOUT remains in the suspended state, as shown in ***Figure 5-8. HXTAL clock source in bypass mode***. The CK_HXTAL is equal to the external clock which drives the OSCIN pin.

Figure 5-8. HXTAL clock source in bypass mode



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.

Internal 48M RC oscillators (IRC48M)

The internal 48M RC oscillator, IRC48M, has a fixed frequency of 48 MHz. The IRC48M oscillator provides a lower cost type clock source as no external components are required when PLLUSB/USBHS used. The IRC48M RC oscillator can be switched on or off using the IRC48MEN bit in the RCU_ADDCTL Register. The IRC48MSTB flag in the RCU_ADDCTL register is used to indicate if the internal 48M RC oscillator is stable. An interrupt can be generated if the related interrupt enable bit, IRC48MSTBIE, in the RCU_ADDINT register, is set when the IRC48M becomes stable. The IRC48M clock is used for the clocks of PLLUSB/USBHS.

The frequency accuracy of the IRC48M can be calibrated by the manufacturer, but its operating frequency is still not enough accurate because the USB need the frequency must between 48MHz with 500ppm accuracy. A hardware automatically dynamic trim performed in CTC unit adjust the IRC48M to the needed frequency.

Phase locked loop (PLL)

There are four internal Phase Locked Loop, the PLL, PLL1, PLL2, and PLLUSB.

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 PLLUSB can be switched on or off by using the PLLUSBEN bit in the RCU_ADDCTL Register. The PLLUSBSTB flag in the RCU_ADDCTL Register will indicate if the PLLUSB clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLUSBSTBIE, in the RCU_ADDINT register, is set as the PLLUSB becomes stable.

The four 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 rnable 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®-M33. 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 RTC, the HXTAL failure will reset the RTC clock selection.

Clock output capability

The clock output capability is ranging from 0.09375 MHz to 180 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

Clock source 0 selection bits	Clock source
1100	CK_IRC48M
1101	CK_IRC48M/8
1110	CK_PLLUSB/32

Voltage control

The 1.1V domain voltage in Deep-sleep mode can be controlled by DSLPVS[2:0] bit in the Deep-sleep mode voltage register (RCU_DSV).

Table 5-4. 1.1V domain voltage selected in deep-sleep mode

DSLPVS[2:0]	Deep-sleep mode voltage(V)
000	1.0
001	0.9
010	0.8
011	0.7

5.6. Register definition

RCU base address: 0x4002 1000

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 4 ~ 32 MHz crystal oscillator (HXTAL) clock monitor 1: Enable the High speed 4 ~ 32 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 ~ 32 MHz crystal oscillator disabled 1: High speed 4 ~ 32 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%.

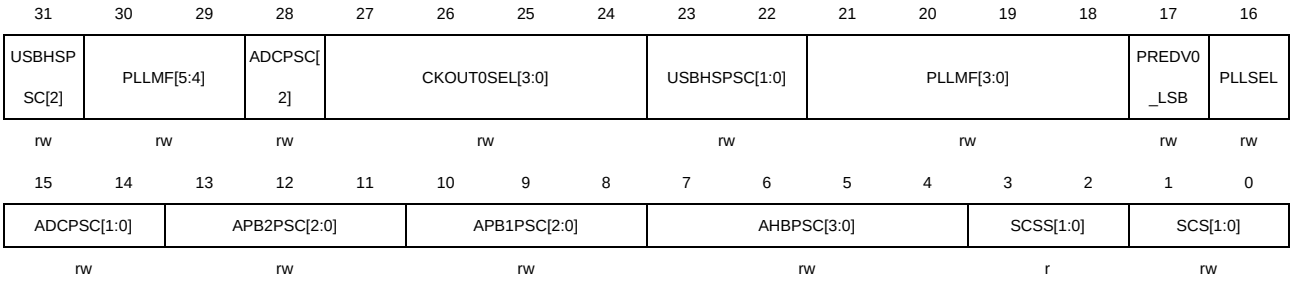
2	Reserved	Must be kept at reset value.
1	IRC8MSTB	IRC8M internal 8MHz RC oscillator stabilization flag Set by hardware to indicate if the IRC8M oscillator is stable and ready for use. 0: IRC8M oscillator is not stable 1: IRC8M oscillator is stable
0	IRC8MEN	Internal 8MHz RC oscillator enable Set and 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).



Bits	Fields	Descriptions
31	USBHSPSC[2]	Bit 2 of USBHSPSC see bits 23:22 of RCU_CFG0
30:29	PLLMF[5:4]	Bit 5 and 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: Internal 8MHz RC 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 1100: CK_IRC48M clock selected 1101: (CK_IRC48M / 8) clock selected 1110: (CK_PLLUSB / 32) clock selected
23:22	USBHSPSC[1:0]	USBHS clock prescaler selection Set and reset by software to control the USBHS clock prescaler value. The USBHS clock must be 48MHz. These bits can't be reset if the USBHS clock is enabled. 000: CK_USBHS = CK_PLL / 1.5 001: CK_USBHS = CK_PLL 010: CK_USBHS = CK_PLL / 2.5 011: CK_USBHS = CK_PLL / 2 100: CK_USBHS = CK_PLL / 3 101: CK_USBHS = CK_PLL / 3.5 11x :CK_USBHS = CK_PLL / 4
21:18	PLLMF[3:0]	The PLL clock multiplication factor Bit 29, bit 30 of RCU_CFG0 and these bits are written by software to define the PLL multiplication factor **Note:** The PLL output frequency must not exceed 180 MHz 000000: (PLL source clock x 2) 000001: (PLL source clock x 3) 000010: (PLL source clock x 4) 000011: (PLL source clock x 5) 000100: (PLL source clock x 6) 000101: (PLL source clock x 7) 000110: (PLL source clock x 8) 000111: (PLL source clock x 9) 001000: (PLL source clock x 10) 001001: (PLL source clock x 11) 001010: (PLL source clock x 12) 001011: (PLL source clock x 13) 001100: (PLL source clock x 14) 001101: (PLL source clock x 6.5) 001110: (PLL source clock x 16) 001111: (PLL source clock x 16) 010000: (PLL source clock x 17) 010001: (PLL source clock x 18) 010010: (PLL source clock x 19) 010011: (PLL source clock x 20) 010100: (PLL source clock x 21) 010101: (PLL source clock x 22) 010110: (PLL source clock x 23)

		010111: (PLL source clock x 24) 011000: (PLL source clock x 25) 011001: (PLL source clock x 26) 011010: (PLL source clock x 27) 011011: (PLL source clock x 28) 011100: (PLL source clock x 29) 011101: (PLL source clock x 30) 011110: (PLL source clock x 31) 011111: (PLL source clock x 32) 100000: (PLL source clock x 33) 100001: (PLL source clock x 34) … 111110: (PLL source clock x 63) 111111: (PLL source clock x 64)
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 clock of PLL 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 or IRC48M (PLLPRESEL of RCU_CFG1 register) selected as source clock of PLL
15:14	ADCPSC[1:0]	ADC clock prescaler selection These bits, bit 28 of RCU_CFG0 and bit 29 of RCU_CFG1 are written by software to define the ADC prescaler factor.Set and cleared by software. 0000: (CK_APB2 / 2) selected 0001: (CK_APB2 / 4) selected 0010: (CK_APB2 / 6) selected 0011: (CK_APB2 / 8) selected 0100: (CK_APB2 / 2) selected 0101: (CK_APB2 / 12) selected 0110: (CK_APB2 / 8) selected 0111: (CK_APB2 / 16) selected 1x00: (CK_AHB / 5) selected 1x01: (CK_AHB / 6) selected 1x10: (CK_AHB / 10) selected 1x11: (CK_AHB / 20) 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. 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 Set and reset by software to enable/disable LXTAL stabilization interrupt 0: Disable the LXTAL stabilization interrupt 1: Enable the LXTAL stabilization interrupt
8	IRC40KSTBIE	IRC40K stabilization interrupt enable Set and reset by software to enable/disable IRC40K stabilization interrupt 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 4 ~ 32 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
CMPRST	Reserved	SHRTIMERRST	USART5RST	Reserved						TIMER10RST	TIMER9RST	TIMER8RST	Reserved		
rw		rw	rw							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	CMPRST	CMP reset This bit is set and reset by software. 0: No reset 1: Reset the CMP
30	Reserved	Must be kept at reset value.
29	SHRTIMERRST	SHRTIMER reset This bit is set and reset by software. 0: No reset 1: Reset the SHRTIMER
28	USART5RST	USART5 reset This bit is set and reset by software. 0: No reset 1: Reset the USART5
27:22	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

5.6.5. 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	I2C2RST	Reserved	I2C1RST	I2C0RST	UART4RST	UART3RST	USART2RST	USART1RST	Reserved
		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
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 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	I2C2RST	I2C2 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C2
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
10:9	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

5.6.6. 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
SQPIEN	TMUEN	Reserved													ENETRXEN
rw	rw														rw

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

rw rw rw rw rw rw rw rw rw rw rw

Bits	Fields	Descriptions
31	SQPIEN	SQPI clock enable This bit is set and reset by software. 0: Disabled SQPI clock 1: Enabled SQPI clock
30	TMUEN	TMUEN clock enable This bit is set and reset by software. 0: Disabled TMUEN clock 1: Enabled TMUEN clock
29:17	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	ULPIEN	ULPI clock enable This bit is set and reset by software. 0: Disabled ULPI clock 1: Enabled ULPI clock
12	USBHSEN	USBHS clock enable This bit is set and reset by software. 0: Disabled USBHS clock 1: Enabled USBHS 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
CMPEN	Reserved	SHRTIMEREN	USART5EN	Reserved						TIMER10EN	TIMER9EN	TIMER8EN	Reserved		
rw		rw	rw							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	CMPEN	CMP clock enable

		This bit is set and reset by software. 0: Disabled CMP clock 1: Enabled CMP clock
30	Reserved	Must be kept at reset value
29	SHRTIMEREN	SHRTIMER clock enable This bit is set and reset by software. 0: Disabled SHRTIMER clock 1: Enabled SHRTIMER clock
28	USART5EN	USART5 clock enable This bit is set and reset by software. 0: Disabled USART5 clock 1: Enabled USART5 clock
27:22	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

5.6.8. 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	I2C2EN	Reserved	I2C1EN	I2C0EN	UART4EN	UART3EN	USART2EN	USART1EN	Reserved
		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
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	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	I2C2EN	I2C2 clock enable This bit is set and reset by software. 0: Disabled I2C2 clock 1: Enabled I2C2 clock
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
10:9	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

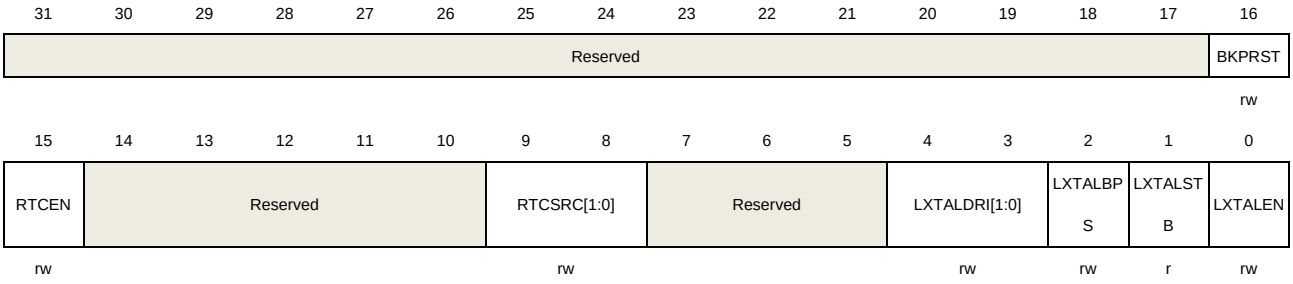
2	TIMER3EN	TIMER3 clock enable This bit is set and reset by software. 0: Disabled TIMER3 clock 1: Enabled TIMER3 clock
1	TIMER2EN	TIMER2 clock enable This bit is set and reset by software. 0: Disabled TIMER2 clock 1: Enabled TIMER2 clock
0	TIMER1EN	TIMER1 clock enable This bit is set and reset by software. 0: Disabled TIMER1 clock 1: Enabled TIMER1 clock

5.6.9. 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.



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
7:5	Reserved	Must be kept at reset value.
4:3	LXTALDRI[1:0]	LXTAL drive capability Set and reset by software. Backup domain reset resets this value. 00: Lower driving capability 01: Medium low driving capability 10: Medium high driving capability 11: Higher driving capability (reset value) **Note:** The LXTALDRI is not in bypass mode.
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														IRC40KSTB	IRC40KEN
														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
SQPIRST	TMURST	Reserved													
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ENETRST	Reserved	USBHRST	Reserved											
	rw		rw												

Bits	Fields	Descriptions
31	SQPIRST	SQPI reset This bit is set and reset by software. 0: No reset 1: Reset the SQPI
30	TMURST	TMU reset This bit is set and reset by software. 0: No reset 1: Reset the TMU
29:15	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	USBHSRST	USBHS reset This bit is set and reset by software. 0: No reset 1: Reset the USBHS
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
PLL2MF[4]	PLLREPSEL	ADCPSC[3]	PLL2MF[5]	Reserved								SHRTIMERSEL	I2S2SEL	I2S1SEL	PREDV0SEL
rw	rw	rw	rw									rw	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	PLL2MF[4]	Bit 4 of PLL2MF see bits 15:12 of RCU_CFG1
30	PLLPRESEL	PLL clock source preselection 0: HXTAL selected as PLL source clock 1: CK_IRC48M selected as PLL source clock
29	ADCPSC[3]	Bit 3 of ADCPSC see bits 15:14 of RCU_CFG0 and bit 28 of RCU_CFG0
28	PLL2MF[5]	Bit 5 of PLL2MF see bits 15:12 of RCU_CFG1
27:20	Reserved	Must be kept at reset value.
19	SHRTIMERSEL	SHRTIMER Clock Source Selection Set and reset by software to control the SHRTIMER clock source. 0: APB2 clock selected as SHRTIMER source clock 1: System clock selected as SHRTIMER source clock
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 or IRC48M 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 These bits, bit 31 and bit28 of RCU_CFG1 are written by software to define the PLL2 multiplication factor. 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: Reserve 1110 :(PLL2 source clock x 16) 1111: (PLL2 source clock x 20) 10000: (PLL2 source clock x 18) … 10110: (PLL2 source clock x 24) 10111: (PLL2 source clock x 25) 11000 :(PLL2 source clock x 26) 11001: (PLL2 source clock x 27) 11010: (PLL2 source clock x 28) 11011: (PLL2 source clock x 29) 11100: (PLL2 source clock x 30) 11101: (PLL2 source clock x 31) 11110 :(PLL2 source clock x 32) 11111: (PLL2 source clock x 40) 100000: (PLL2 source clock x 34) … 111110 :(PLL2 source clock x 64) 111111: (PLL2 source clock x 80)
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: reserve
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[1:0]	
														rw	

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2:0	DSLPVS[1:0]	Deep-sleep mode voltage select These bits are set and reset by software 00: The core voltage is 1.0V in Deep-sleep mode 01: The core voltage is 0.9V in Deep-sleep mode 10: The core voltage is 0.8V in Deep-sleep mode 11: The core voltage is 0.7V in Deep-sleep mode

5.6.14. Additional clock control register (RCU_ADDCTL)

Address offset: 0xC0
Reset value: 0x8000 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
IRC48MCALIB[7:0]								Reserved						IRC48MSTB	IRC48MEN
r														r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLUSBSTB	PLLUSBEN	Reserved							USBSWEN	USBHSDV[2:0]			USBHSSEL	CK48MSEL[1:0]	
r	rw								rw	rw			rw	rw	

Bits	Fields	Descriptions
31:24	IRC48MCALIB [7:0]	Internal 48MHz RC oscillator calibration value register These bits are load automatically at power on.
23:18	Reserved	Must be kept at reset value.
17	IRC48MSTB	Internal 48MHz RC oscillator clock stabilization flag Set by hardware to indicate if the IRC48M oscillator is stable and ready for use. 0: IRC48M is not stable 1: IRC48M is stable
16	IRC48MEN	Internal 48MHz RC oscillator enable Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: IRC48M disable 1: IRC48M enable
15	PLLUSBSTB	PLLUSB clock stabilization flag Set by hardware to indicate if the PLLUSB clock is stable and ready for use. 0: PLLUSB clock is not stable 1: PLLUSB clock is stable
14	PLLUSBEN	PLLUSB enable This bit is set and reset by software 0: PLLUSB disable 1: PLLUSB enable
13:7	Reserved	Must be kept at reset value.
6	USBSWEN	USB clock source selection enable 0: hardware switch USB clock by USBHS module 1: use USBSW to switch USB clock
5:3	USBHSDV[2:0]	USBHS clock divider factor This bit is set and reset by software. 000: USBHSDV input source clock divided by 2 001: USBHSDV input source clock divided by 4 ... 111: USBHSDV input source clock divided by 16
2	USBHSSEL	USBHS clock source selection Set and reset by software to control the USBHS clock source. 0: 48M clock selected as USBHS source clock 1: 60M clock selected as USBHS source clock
1:0	CK48MSEL[1:0]	USB 48M clock source selection 00: not effect (select PLL/USBPRE) 01: CK_IRC48M select as CK48M clock source 10: CK_PLLUSB/USBHSDV select as CK48M clock source

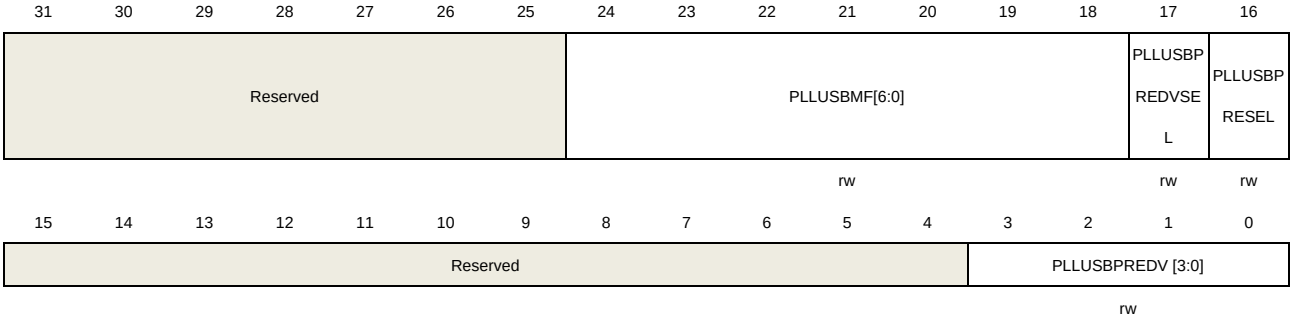
11: CK_PLL2 select as CK48M clock source

5.6.15. Additional Clock configuration register (RCU_ADDCFG)

Address offset: 0XC4

Reset value: 0x0000 0000

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



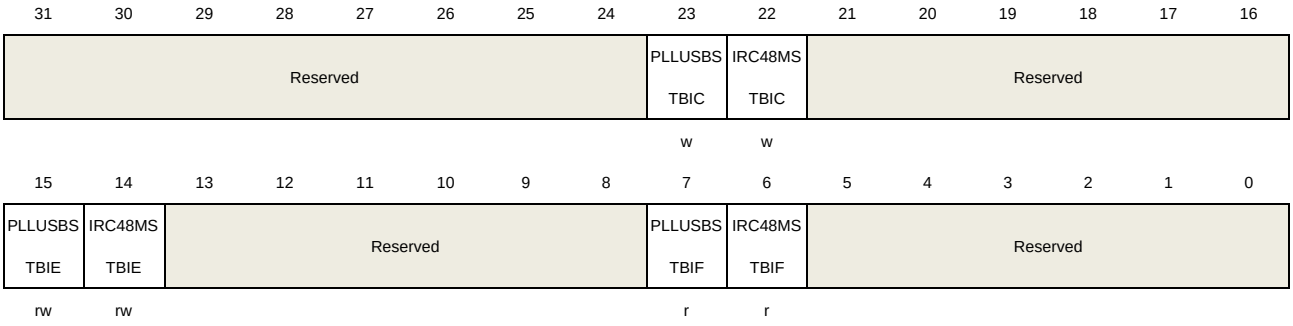
Bits	Fields	Descriptions
31:25	Reserved	Must be kept at reset value.
24:18	PLLUSBMF[6:0]	The PLLUSB clock multiplication factor **Note:** The PLLUSB output frequency must not exceed 480 MHz 0000000~0001111: Reserved 0010000: CK_PLLUSB = CK_PLLUSBSRC x 16 0010001: CK_PLLUSB = CK_PLLUSBSRC x 17 0010010: CK_PLLUSB = CK_PLLUSBSRC x 18 0010011: CK_PLLUSB = CK_PLLUSBSRC x 19 … 1111111: CK_PLLUSB = CK_PLLUSBSRC x127
17	PLLUSBPREDVSEL	PLLUSBPREDV input clock source selection Set and reset by software. 0: PLLUSBSRC output selected as PLLUSBPREDV input source clock 1: CK_PLL1 output clock selected as PLLUSBPREDV input source clock
16	PLLUSBPRESEL	PLLUSB clock source preselection Set and reset by software to control the PLLUSB clock source. 0: CK_ HXTAL selected as PLLUSB source clock 1: CK_IRC48M output clock selected as PLLUSB source clock
15:4	Reserved	Must be kept at reset value.
3:0	PLLUSBPREDV[3:0]	PLLUSBPREDV division factor This bit is set and reset by software. 0000: Reserved 0001: PLLUSBPREDV input source clock divided by 1

0010: PLLUSBPREDV input source clock divided by 2
0011: PLLUSBPREDV input source clock divided by 3
0100: PLLUSBPREDV input source clock divided by 4
0101: PLLUSBPREDV input source clock divided by 5
0110: PLLUSBPREDV input source clock divided by 6
0111: PLLUSBPREDV input source clock divided by 7
1000: PLLUSBPREDV input source clock divided by 8
1001: PLLUSBPREDV input source clock divided by 9
1010: PLLUSBPREDV input source clock divided by 10
1011: PLLUSBPREDV input source clock divided by 11
1100: PLLUSBPREDV input source clock divided by 12
1101: PLLUSBPREDV input source clock divided by 13
1110: PLLUSBPREDV input source clock divided by 14
1111: PLLUSBPREDV input source clock divided by 15

5.6.16. Additional clock interrupt register (RCU_ADDINT)

Address offset: 0xCC
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23	PLLUSBSTBIC	PLLUSB stabilization interrupt clear Write 1 by software to reset the PLLUSBSTBIF flag. 0: Not reset PLLUSBSTBIF flag 1: Reset PLLUSBSTBIF flag
22	IRC48MSTBIC	Internal 48 MHz RC oscillator Stabilization interrupt clear Write 1 by software to reset the IRC48MSTBIF flag. 0: Not reset IRC48MSTBIF flag 1: Reset IRC48MSTBIF flag
21:16	Reserved	Must be kept at reset value.
15	PLLUSBSTBIE	PLLUSB stabilization interrupt enable

		Set and reset by software to enable/disable the PLLUSB stabilization interrupt 0: Disable the PLLUSB stabilization interrupt 1: Enable the PLLUSB stabilization interrupt
14	IRC48MSTBIE	Internal 48 MHz RC oscillator stabilization interrupt enable Set and reset by software to enable/disable the IRC48M stabilization interrupt 0: Disable the IRC48M stabilization interrupt 1: Enable the IRC48M stabilization interrupt
13:8	Reserved	Must be kept at reset value.
7	PLLUSBSTBIF	PLLUSB stabilization interrupt flag Set by hardware when the PLLUSB clock is stable and the PLLUSBSTBIE bit is set. Reset by software when setting the PLLUSBTBIC bit. 0: No PLLUSB stabilization interrupt generated 1: PLLUSB stabilization interrupt generated
6	IRC48MSTBIF	IRC48M stabilization interrupt flag Set by hardware when the Internal 48 MHz RC oscillator clock is stable and the IRC48MSTBIE bit is set. Reset by software when setting the IRC48MSTBIC bit. 0: No IRC48M stabilization interrupt generated 1: IRC48M stabilization interrupt generated
5:0	Reserved	Must be kept at reset value.

5.6.17. PLL clock spread spectrum control register (RCU_PLLSSCTL)

Address offset: 0XD0
Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit).
The spread spectrum modulation is available only for the main PLL clock
The RCU_PLLSSCTL register must be written when the main PLL is disabled.
This register is used to configure the PLL spread spectrum clock generation according to the following formulas:

MODCNT = round(f_{PLLIN}/4/f_{mod})

MODSTEP = round(mdamp*PLLN*2^{15}/(MODCNT*100))

Where f_{PLLIN} represents the PLL input clock frequency, f_{mod} represents the spread spectrum modulation frequency, mdamp represents the spread spectrum modulation amplitude expressed as a percentage, PLLN represents the PLL clock frequency multiplication factor.

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

MODSTEP[2:0]	MODCNT[12:0]
rw	rw

Bits	Fields	Descriptions
31	SSCGON	PLL spread spectrum modulation enable 0: Spread spectrum modulation disable 1: Spread spectrum modulation enable
30	SS_TYPE	PLL spread spectrum modulation type select 0: Center spread selected 1: Down spread selected
29:28	Reserved	Must be kept at reset value.
27:13	MODSTEP[14:0]	These bits configure PLL spread spectrum modulation profile amplitude and frequency. The following criteria must be met: MODSTEP*MODCNT≤2^{15}-1
12:0	MODCNT[12:0]	These bits configure PLL spread spectrum modulation profile amplitude and frequency. The following criteria must be met: MODSTEP*MODCNT≤2^{15}-1

5.6.18. Clock configuration register 2 (RCU_CFG2)

Address offset: 0xD4
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										I2C2SEL[1:0]		Reserved		USART5SEL[1:0]	
										rw				rw	

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5:4	I2C2SEL[1:0]	I2C2 Clock Source Selection Set and reset by software to control the I2C2 clock source. 00: APB1 clock selected as I2C2 source clock 01: System clock selected as I2C2 source clock 1x: CK_IRC8M clock selected as I2C2 source clock
3:2	Reserved	Must be kept at reset value.
1:0	USART5SEL[1:0]	USART5 clock source selection Set and reset by software to control the USART5 clock source. 00: CK_APB2 selected as USART5 source clock

01: CK_SYS selected as USART5 source clock

10: CK_LXTAL selected as USART5 source clock

11: CK_IRC8M selected as USART5 source clock

5.6.19. APB1 additional reset register (RCU_ADDAPB1RST)

Address offset: 0xE0

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
CAN2RST	Reserved			CTCRST	Reserved										
rw				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Bits	Fields	Descriptions
31	CAN2RST	CAN2 reset This bit is set and reset by software. 0: No reset 1: Reset CAN2
30:28	Reserved	Must be kept at reset value.
27	CTCRST	CTC reset This bit is set and reset by software. 0: No reset 1: Reset CTC
26:0	Reserved	Must be kept at reset value.

5.6.20. APB1 additional enable register (RCU_ADDAPB1EN)

Address offset: 0xE4

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
CAN2EN	Reserved			CTCEN	Reserved										
rw				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Bits	Fields	Descriptions
31	CAN2EN	CAN2 enable This bit is set and reset by software. 0: Disabled CAN clock 1: Enabled CAN clock
30:28	Reserved	Must be kept at reset value.
27	CTCEN	CTC clock enable This bit is set and reset by software. 0: Disabled CTC clock 1: Enabled CTC clock
26:0	Reserved	Must be kept at reset value.

6. Clock trim controller (CTC)

6.1. Overview

The Clock Trim Controller (CTC) is used to trim internal 48MHz RC oscillator (IRC48M) automatically by hardware. The CTC unit trim the frequency of the IRC48M based on an external accurate reference signal source. It can automatically adjust the trim value to provide a precise IRC48M clock.

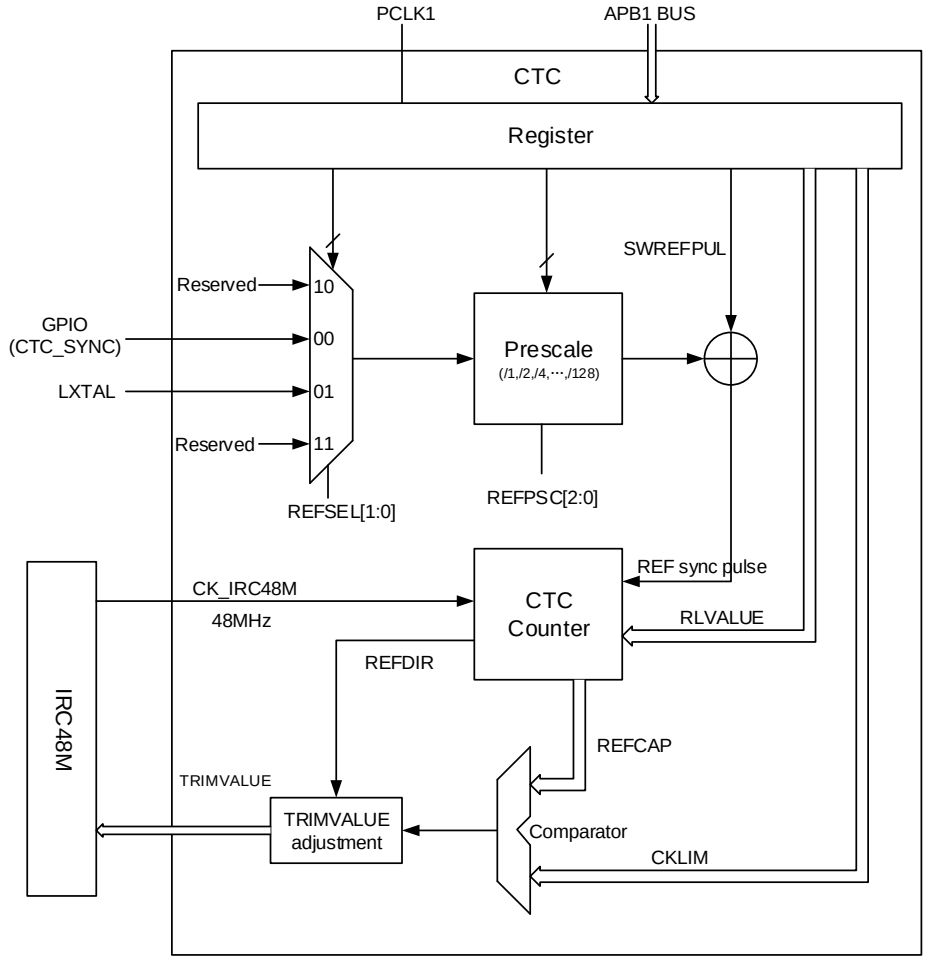
6.2. Characteristics

- Two external reference signal source: GPIO(CTC_SYNC), LXTAL clock.
- Provide software reference sync pulse.
- Automatically trimmed by hardware without any software action.
- 16 bits trim counter with reference signal source capture and reload.
- 8 bits clock trim base value to frequency evaluation and automatically trim.
- Enough flag or interrupt to indicate the clock is OK (CKOKIF), warning (CKWARNIF) or error (ERRIF).

6.3. Function overview

Figure 6-1. CTC overview provides details on the internal configuration of the CTC.

Figure 6-1. CTC overview



6.3.1. REF sync pulse generator

Firstly, the reference signal source can select GPIO(CTC_SYNC) or LXTAL clock by setting REFSEL bits in CTC_CTL1 register.

Secondly, the selected reference signal source use a configurable polarity by setting REFPOL bit in CTC_CTL1 register, and can be divided to a suitable frequency with a configurable prescaler by setting REFPSC bits in CTC_CTL1 register.

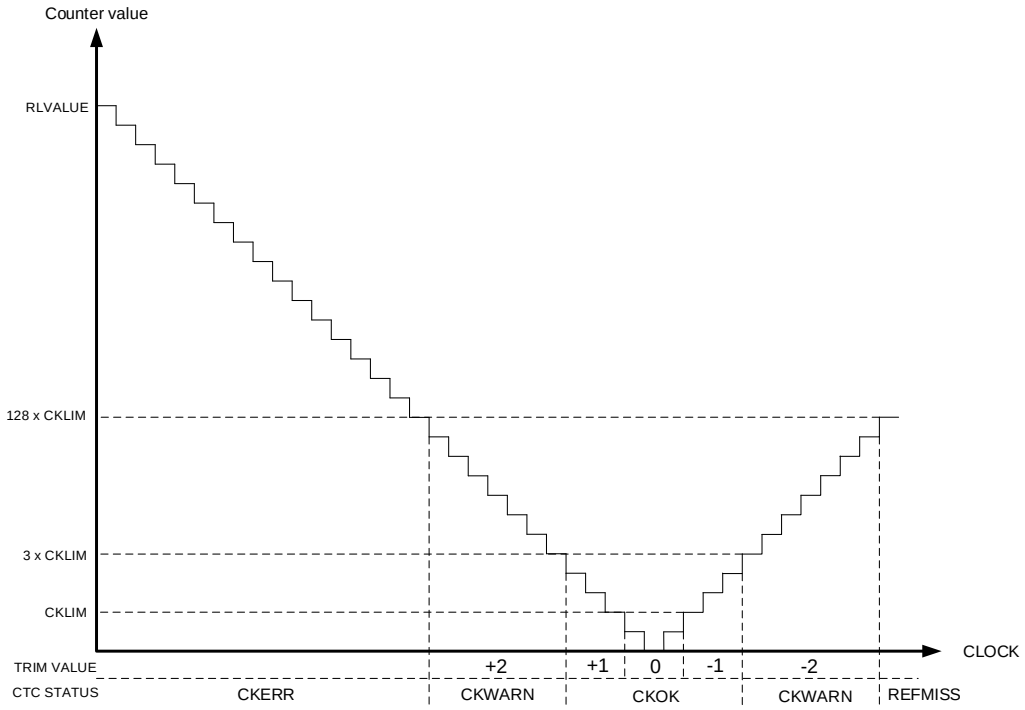
Thirdly, if a software reference pulse needed, write 1 to SWREFPUL bit in CTC_CTL0 register. The software reference pulse generated in last step is logical OR with the external reference pulse.

6.3.2. CTC trim counter

The CTC trim counter is clocked by CK_IRC48M. After CNTEN bit in CTC_CTL0 register set, and a first REF sync pulse detected, the counter start down-counting from RLVALUE (defined in CTC_CTL1 register). If any REF sync pulse detected, the counter reload the RLVALUE and start down-counting again. If no REF sync pulse detected, the counter down-count to zero,

and then up- counting to 128 x CKLIM (defined in CTC_CTL1 register), and then stop until next REF sync pulse detected. If any REF sync pulse detected, the current CTC trim counter value is captured to REFCAP in status register (CTC_STAT), and the counter direction is captured to REFDIR in status register (CTC_STAT). The detail is showing in ***Figure 6-2. CTC trim counter***.

Figure 6-2. CTC trim counter



6.3.3. Frequency evaluation and automatically trim process

The clock frequency evaluation is performed when a REF sync pulse occur. If a REF sync pulse occurs on down-counting, it means the current clock is slower than correct clock (the frequency of 48M).It needs to improve TRIMVALUE in CTC_CTL0 register. If a REF sync pulse occurs on up-counting, it means the current clock is faster than correct clock (the frequency of 48M).It needs to reduce TRIMVALUE in CTC_CTL0 register. The CKOKIF, CKWARNIF, CKERR and REFMISS in CTC_STAT register shows the frequency evaluation scope.

If the AUTOTRIM bit in CTC_CTL0 register is setting, the automatically hardware trim mode enabled. In this mode, if a REF sync pulse occurs on down-counting, it means the current clock is slower than correct clock, the TRIMVALUE will be increased automatically to raise the clock frequency. Vice versa when it occurs on up-counting, the TRIMVALUE will be reduced automatically to reduce the clock frequency.

- Counter < CKLIM when REF sync pulse is detected.

  The CKOKIF in CTC_STAT register set, and an interrupt generated if CKOKIE bit in CTC_CTL0 register is 1.

If the AUTOTRIM bit in CTC_CTL0 register set, the TRIMVALUE in CTC_CTL0 register is not changed.

- CKLIM ≤ Counter < 3 x CKLIM when REF sync pulse is detected.

  The CKOKIF in CTC_STAT register set, and an interrupt generated if CKOKIE bit in CTC_CTL0 register is 1.

  If the AUTOTRIM bit in CTC_CTL0 register set, the TRIMVALUE in CTC_CTL0 register add 1 when down-counting or sub 1 when up-counting.

- 3 x CKLIM ≤ Counter < 128 x CKLIM when REF sync pulse is detected.

  The CKWARNIF in CTC_STAT register set, and an interrupt generated if CKWARNIE bit in CTC_CTL0 register is 1.

  If the AUTOTRIM bit in CTC_CTL0 register set, the TRIMVALUE in CTC_CTL0 register add 2 when down-counting or sub 2 when up-counting.

- Counter ≥ 128 x CKLIM when down-counting when a REF sync pulse is detected.

  The CKERR in CTC_STAT register set, and an interrupt generated if ERRIE bit in CTC_CTL0 register is 1.

  The TRIMVALUE in CTC_CTL0 register is not changed

- Counter = 128 x CKLIM when up-counting.

  The REFMISS in CTC_STAT register set, and an interrupt generated if ERRIE bit in CTC_CTL0 register is 1.

  The TRIMVALUE in CTC_CTL0 register is not changed.

If adjusting the TRIMVALUE in CTC_CTL0 register over the value of 63, the overflow will be occurred, while adjusting the TRIMVALUE under the value of 0, the underflow will be occurred. The TRIMVALUE is in the range 0 to 63 (the TRIMVALUE is 63 if overflow, the TRIMVALUE is 0 if underflow). Then, the TRIMERR in CTC_STAT register will be set, and an interrupt generated if ERRIE bit in CTC_CTL0 register is 1.

6.3.4. Software program guide

The RLVALUE and CKLIM bits in CTC_CTL1 register is critical to evaluate the clock frequency and automatically hardware trim. The value is calculated by the correct clock frequency (IRC48M:48 MHz) and the frequency of REF sync pulse. The ideal case is REF sync pulse occur when the CTC counter is zero, so the RLVALUE is:

$$RLVALUE = (F_{clock} \div F_{REF}) - 1 \quad (6\text{-}1)$$

The CKLIM is set by user according to the clock accuracy. It is recommend to set to the half of the step size, so the CKLIM is:

$$CKLIM = (F_{clock} \div F_{REF}) \times 0.12\% \div 2 \quad (6\text{-}2)$$

The typical step size is 0.12%. Where the F_{clock} is the frequency of correct clock (IRC48M), the F_{REF} is the frequency of reference sync pulse.

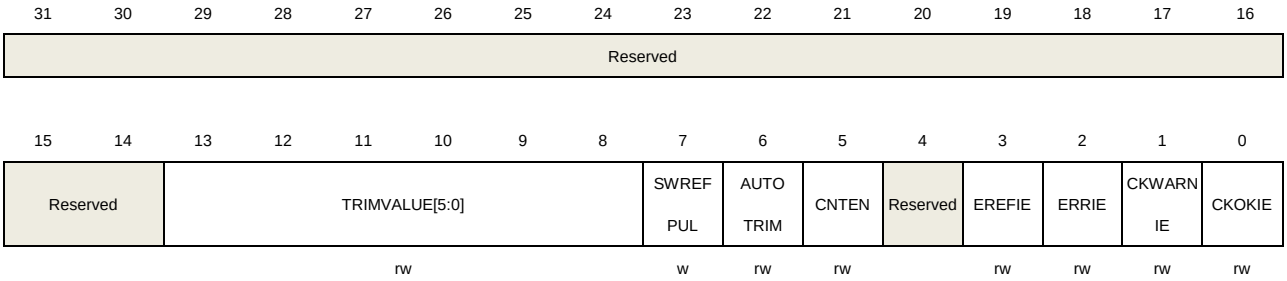
6.4. Register definition

CTC base address: 0x4000 C800

6.4.1. Control register 0 (CTC_CTL0)

Address offset: 0x00
Reset value: 0x0000 2000

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



Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:8	TRIMVALUE[5:0]	IRC48M trim value When AUTOTRIM in CTC_CTL0 register is 0, these bits are set and cleared by software. This mode used to software calibration. When AUTOTRIM in CTC_CTL0 register is 1, these bits are read only. The value automatically modified by hardware. This mode used to hardware trim. The middle value is 32. When increase 1, the IRC48M clock frequency add around 57KHz. When decrease 1, the IRC48M clock frequency sub around 57KHz.
7	SWREFPUL	Software reference source sync pulse This bit is set by software, and generates a reference sync pulse to CTC counter. This bit is cleared by hardware automatically and read as 0. 0: No effect 1: Generates a software reference source sync pulse
6	AUTOTRIM	Hardware automatically trim mode This bit is set and cleared by software. When this bit is set, the hardware automatic trim enabled, the TRIMVALUE bits in CTC_CTL0 register are modified by hardware automatically, until the frequency of IRC48M clock is close to 48MHz. 0: Hardware automatic trim disabled 1: Hardware automatic trim enabled
5	CNTEN	CTC counter enable This bit is set and cleared by software. This bit used to enable or disable the CTC trim counter. When this bit is set, the CTC_CTL1 register cannot be modified.

		0: CTC trim counter disabled 1: CTC trim counter enabled.
4	Reserved	Must be kept at reset value.
3	EREFIE	EREFIF interrupt enable 0: EREFIF interrupt disable 1: EREFIF interrupt enable
2	ERRIE	Error (ERRIF) interrupt enable 0: ERRIF interrupt disable 1: ERRIF interrupt enable
1	CKWARNIE	Clock trim warning (CKWARNIF) interrupt enable 0: CKWARNIF interrupt disable 1: CKWARNIF interrupt enable
0	CKOKIE	Clock trim OK (CKOKIF) interrupt enable 0: CKOKIF interrupt disable 1: CKOKIF interrupt enable

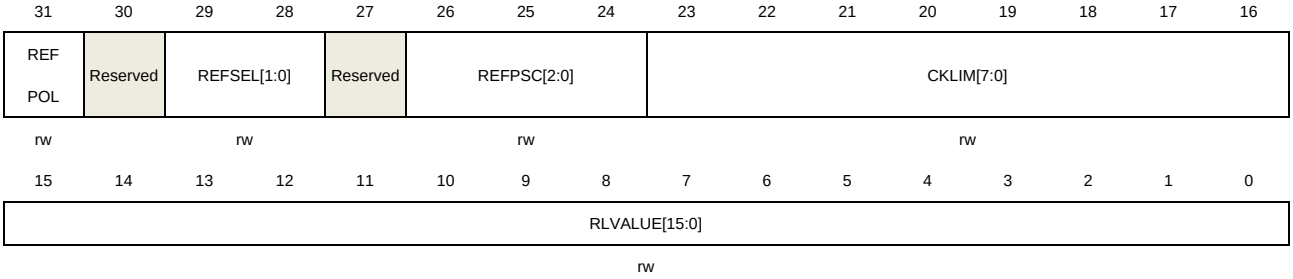
6.4.2. Control register 1 (CTC_CTL1)

Address offset: 0x04

Reset value: 0x2022 BB7F

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

Note: This register cannot be modified when CNTEN is 1.



Bits	Fields	Descriptions
31	REFPOL	Reference signal source polarity This bit is set and cleared by software to select reference signal source polarity 0: Rising edge selected 1: Falling edge selected
30	Reserved	Must be kept at reset value.
29:28	REFSEL[1:0]	Reference signal source selection These bits are set and cleared by software to select reference signal source. 00: GPIO (CTC_SYNC) selected

		01: LXTAL clock selected 10: Reserved. 11: Reserved
27	Reserved	Must be kept at reset value.
26:24	REFPSC[2:0]	Reference signal source prescaler These bits are set and cleared by software 000: Reference signal not divided 001: Reference signal divided by 2 010: Reference signal divided by 4 011: Reference signal divided by 8 100: Reference signal divided by 16 101: Reference signal divided by 32 110: Reference signal divided by 64 111: Reference signal divided by 128
23:16	CKLIM[7:0]	Clock trim base limit value These bits are set and cleared by software to define the clock trim base limit value. These bits used to frequency evaluation and automatically trim process. Please refer to the ***Frequency evaluation and automatically trim process*** for detail.
15:0	RLVALUE[15:0]	CTC counter reload value These bits are set and cleared by software to define the CTC counter reload value. These bits reload to CTC trim counter, when a reference sync pulse is received, so as to start or restart the counter.

6.4.3. Status register (CTC_STAT)

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
REFCAP[15:0]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFDIR	Reserved				TRIM ERR	REF MISS	CKERR	Reserved				EREFIF	ERRIF	CKWARN IF	CKOK IF
r					r	r	r					r	r	r	r

Bits	Fields	Descriptions
31:16	REFCAP[15:0]	CTC counter capture when reference sync pulse. When a reference sync pulse occurred, the CTC trim counter value is captured to REFCAP bits.

15	REFDIR	CTC trim counter direction when reference sync pulse When a reference sync pulse occurred during the counter is working, the CTC trim counter direction is captured to REFDIR bit. 0: Up-counting 1: Down-counting
14:11	Reserved	Must be kept at reset value.
10	TRIMERR	Trim value error bit This bit is set by hardware when the TRIMVALUE in CTC_CTL0 register overflow or underflow. When the ERRIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register. 0: No trim value error occur 1: Trim value error occur
9	REFMISS	Reference sync pulse miss This bit is set by hardware when the reference sync pulse miss. This is occur when the CTC trim counter reach to 128 x CKLIM during up counting and no reference sync pulse detected. This means the clock is too fast to be trimmed to correct frequency or other error occur. When the ERRIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register. 0: No Reference sync pulse miss occur 1: Reference sync pulse miss occur
8	CKERR	Clock trim error bit This bit is set by hardware when the clock trim error occur. This is occur when the CTC trim counter greater or equal to 128 x CKLIM during down counting when a reference sync pulse detected. This means the clock is too slow and cannot be trimmed to correct frequency. When the ERRIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register. 0: No Clock trim error occur 1: Clock trim error occur
7:4	Reserved	Must be kept at reset value.
3	EREFIF	Expect reference interrupt flag This bit is set by hardware when the CTC counter reach to 0. When the EREFIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to EREFIC bit in CTC_INTC register. 0 : No Expect reference occur 1: Expect reference occur
2	ERRIF	Error interrupt flag This bit is set by hardware when an error occurred. If any error of TRIMERR, REFMISS or CKERR occurred, this bit will be set. When the ERRIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register.

0 : No Error occur
1: An error occur

1	CKWARNIF	Clock trim warning interrupt flag This bit is set by hardware when a clock trim warning occurred. If the CTC trim counter greater or equal to 3 x CKLIM and smaller to 128 x CKLIM when a reference sync pulse detected, this bit will be set. This means the clock is too slow or too fast, but can be trim to correct frequency. The TRIMVALUE add 2 or sub 2 when a clock trim warning occurred. When the CKWARNIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to CKWARNIC bit in CTC_INTC register. 0 : No Clock trim warning occur 1: Clock trim warning occur
0	CKOKIF	Clock trim OK interrupt flag This bit is set by hardware when the clock trim is OK. If the CTC trim counter smaller to 3 x CKLIM when a reference sync pulse detected, this bit will be set. This means the clock is OK to use. The TRIMVALUE need not to adjust or adjust one step. When the CKOKIE in CTC_CTL0 register is, an interrupt occur. This bit is cleared by writing 1 to CKOKIC bit in CTC_INTC register. 0 : No Clock trim OK occur 1: Clock trim OK occur

6.4.4. Interrupt clear register (CTC_INTC)

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												EREFIC	ERRIC	CKWARNIC	CKOKIC
												w	w	w	w

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.
3	EREFIC	EREFIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear EREFIF bit in CTC_STAT register. Write 0 is no effect.
2	ERRIC	ERRIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear ERRIF, TRIMERR,

		REFMISS and CKERR bits in CTC_STAT register. Write 0 is no effect.
1	CKWARNIC	CKWARNIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear CKWARNIF bit in CTC_STAT register. Write 0 is no effect.
0	CKOKIC	CKOKIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear CKOKIF bit in CTC_STAT register. Write 0 is no effect.

7. Interrupt/event controller (EXTI)

7.1. Overview

Cortex®-M33 integrates the Nested Vectored Interrupt Controller (NVIC) for efficient exception and interrupts processing. NVIC facilitates low-latency exception and interrupt handling and power management controls. It's tightly coupled to the processer core. You can read the technical reference manual of Cortex®-M33 for more details about NVIC.

EXTI (interrupt / event controller) contains up to 22 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.

7.2. Characteristics

- Cortex®-M33 system exception.
- Up to 88 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 22 independent edge detectors in EXTI.
- Three trigger types: rising, falling and both edges.
- Software interrupt or event trigger.
- Trigger sources configurable.

7.3. Interrupts function overview

The Arm Cortex®-M33 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 ***Table 7-1. NVIC exception types in Cortex®-M33*** and ***Table 7-2. Interrupt vector table*** list all exception types.

Table 7-1. NVIC exception types in Cortex®-M33

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 7-2. 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 and CTC interrupt	RCU and CTC 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
IRQ 13	29	DMA0 channel2 global interrupt	DMA0 channel2 global interrupt	0x0000_0074
IRQ 14	30	DMA0 channel3 global	DMA0 channel3 global	0x0000_0078

Interrupt number	Vector number	Non-connectivity devices interrupt description	Connectivity devices interrupt description	Vector address
		interrupt	interrupt	
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 and TIMER8 global interrupt	0x0000_00A0
IRQ 25	41	TIMER0 update interrupt and TIMER9 global interrupt	TIMER0 update interrupt and TIMER9 global interrupt	0x0000_00A4
IRQ 26	42	TIMER0 trigger and Channel commutation interrupts and TIMER10 global interrupt	TIMER0 trigger and Channel commutation interrupts and TIMER10 global interrupt	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 or I2S1ADD global interrupt	SPI1 or I2S1ADD 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
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

Interrupt number	Vector number	Non-connectivity devices interrupt description	Connectivity devices interrupt description	Vector address
IRQ 42	58	USBD wakeup from EXTI interrupt	USBHS wakeup from EXTI interrupt	0x0000_00E8
IRQ 43	59	TIMER7 break interrupt and TIMER11 global interrupt	TIMER7 break interrupt and TIMER11 global interrupt	0x0000_00EC
IRQ 44	60	TIMER7 update interrupt and TIMER12 global interrupt	TIMER7 update interrupt and TIMER12 global interrupt	0x0000_00F0
IRQ 45	61	TIMER7 trigger and Channel commutation interrupts and TIMER13 global interrupt	TIMER7 trigger and Channel commutation interrupts and TIMER13 global interrupt	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 or I2S2ADD global interrupt	SPI2 or I2S2ADD 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 or DAC global interrupt	TIMER5 or DAC 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	CAN1 TX interrupt	CAN1 TX interrupt	0x0000_013C
IRQ64	80	CAN1 RX0 interrupt	CAN1 RX0 interrupt	0x0000_0140
IRQ65	81	CAN1 RX1 interrupt	CAN1 RX1 interrupt	0x0000_0144
IRQ66	82	CAN1 EWMC interrupt	CAN1 EWMC interrupt	0x0000_0148
IRQ67	83	reserved	USBHS global interrupt	0x0000_014C
IRQ68	84	reserved	reserved	0x0000_0150

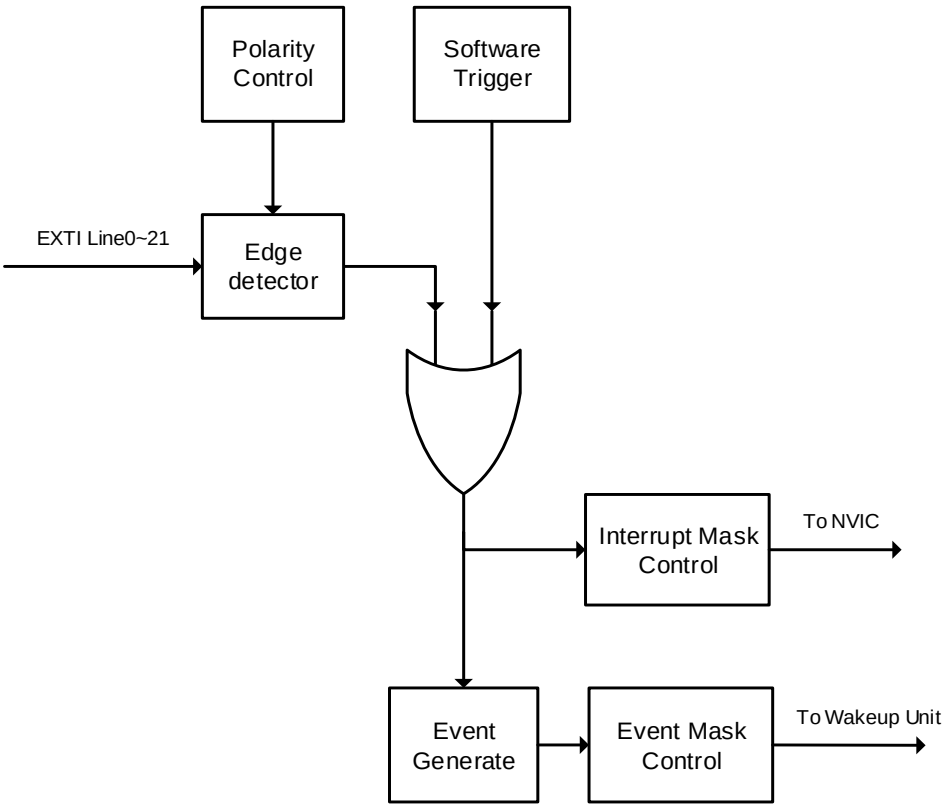
Interrupt number	Vector number	Non-connectivity devices interrupt description	Connectivity devices interrupt description	Vector address
IRQ69	85	SHRTIMER_IRQ2 interrupt	SHRTIMER_IRQ2 interrupt	0x0000_0154
IRQ70	86	SHRTIMER_IRQ3 interrupt	SHRTIMER_IRQ3 interrupt	0x0000_0158
IRQ71	87	SHRTIMER_IRQ4 interrupt	SHRTIMER_IRQ4 interrupt	0x0000_015C
IRQ72	88	SHRTIMER_IRQ5 interrupt	SHRTIMER_IRQ5 interrupt	0x0000_0160
IRQ73	89	SHRTIMER_IRQ6 interrupt	SHRTIMER_IRQ6 interrupt	0x0000_0164
IRQ74	90	reserved	USBHS_EP1_OUT interrupt	0x0000_0168
IRQ75	91	reserved	USBHS_EP1_IN interrupt	0x0000_016C
IRQ76	92	SHRTIMER_IRQ0 interrupt	SHRTIMER_IRQ0 interrupt	0x0000_0170
IRQ77	93	SHRTIMER_IRQ1 interrupt	SHRTIMER_IRQ1 interrupt	0x0000_0174
IRQ78	94	reserved	CAN2_TX interrupt	0x0000_0178
IRQ79	95	reserved	CAN2_RX0 interrupt	0x0000_017C
IRQ80	96	reserved	CAN2_RX1 interrupt	0x0000_0180
IRQ81	97	reserved	CAN2_EWMC interrupt	0x0000_0184
IRQ82	98	I2C2_EV interrupt	I2C2_EV interrupt	0x0000_0188
IRQ83	99	I2C2_ER interrupt	I2C2_ER interrupt	0x0000_018C
IRQ84	100	USART5 global interrupt	USART5 global interrupt	0x0000_0190
IRQ85	101	I2C2 Wakeup from EXTI interrupt	I2C2 Wakeup from EXTI interrupt	0x0000_0194
IRQ86	102	USART5 Wakeup from EXTI interrupt	USART5 Wakeup from EXTI interrupt	0x0000_0198
IRQ87	103	reserved	TMU interrupt	0x0000_019C

Note:

At non-connectivity devices, USB and CAN (IRQ19, IRQ 20) function cannot be used at the same time.

7.4. External interrupt and event (EXTI) block diagram

Figure 7-1. Block diagram of EXTI



7.5. External Interrupt and event function overview

The EXTI contains up to 22 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 6 lines from internal modules which refers to ***Table 7-3. EXTI source*** for detail. All GPIO pins can be selected as an EXTI trigger source by configuring AFIO_EXTISSx registers in GPIO module (please refer to ***General-purpose and alternate-function I/Os (GPIO and AFIO)*** section for detail).

EXTI can provide not only interrupts but also event signals to the processor. The Cortex®-M33 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.

7.5.1. Hardware trigger

Hardware trigger may be used to detect the voltage change of external or internal signals. The software should follow these steps to use this function:

1. Configure EXTI sources in AFIO module based on application requirement.
2. Configure EXTI_RTEN and EXTI_FTEN to enable the rising or falling detection on related pins. (Software may set both RTENx and FTENx for a pin at the same time to detect both rising and falling changes on this pin).
3. Enable interrupts or events by setting related EXTI_INTEN or EXTI_EVEN bits.
4. EXTI starts to detect changes on the configured pins. The related interrupt or event will be triggered when desired change is detected on these pins. If the interrupt is triggered, the related PDx is set; if the event is triggered, the related PDx is not set. The software should response to the interrupts or events and clear these PDx bits.

7.5.2. Software trigger

Software may also trigger EXTI interrupts or events following these steps:

1. Enable interrupts or events by setting related EXTI_INTEN or EXTI_EVEN bits.
2. Set SWIEVx bits in EXTI_SWIEV register, the related interrupt or event will be triggered immediately. If the interrupt is triggered, the related PDx is set; if the event is triggered, the related PDx is not set. Software should response to these interrupts, and clear related PDx bits.

Table 7-3. 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

EXTI Line Number	Source
16	LVD
17	RTC Alarm
18	USB Wakeup
19	Ethernet Wakeup
20	I2C2 Wakeup
21	USART5 Wakeup

7.6. EXTI Register

EXTI base address: 0x4001 0400

7.6.1. 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										INTEN21	INTEN20	INTEN19	INTEN18	INTEN17	INTEN16
										rw	rw	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
31:22	Reserved	Must be kept at reset value.
21: 0	INTENx	Interrupt enable bit x (x = 0…21) 0: Interrupt from linex is disabled 1: Interrupt from linex is enabled

7.6.2. 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										EVEN21	EVEN20	EVEN19	EVEN18	EVEN17	EVEN16
										rw	rw	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
31:22	Reserved	Must be kept at reset value.
21: 0	EVENx	Event enable bit x (x = 0…21) 0: Event from linex is disabled. 1: Event from linex is enabled.

7.6.3. 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										RTEN21	RTEN20	RTEN19	RTEN18	RTEN17	RTEN16
										rw	rw	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
31:22	Reserved	Must be kept at reset value.
21:0	RTENx	Rising edge trigger enable x (x = 0…21) 0: Rising edge of linex is invalid 1: Rising edge of linex is valid as an interrupt / event request

7.6.4. 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										FTEN21	FTEN20	FTEN19	FTEN18	FTEN17	FTEN16
										rw	rw	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: 22	Reserved	Must be kept at reset value.
21: 0	FTENx	Falling edge trigger enable x (x = 0…21) 0: Falling edge of linex is invalid 1: Falling edge of linex is valid as an interrupt / event request

7.6.5. 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										SWIEV21	SWIEV20	SWIEV19	SWIEV18	SWIEV17	SWIEV16
										rw	rw	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
31:22	Reserved	Must be kept at reset value.
21: 0	SWIEVx	Interrupt / Event software trigger x (x = 0…21) 0: Deactivate the EXTIx software interrupt / event request 1: Activate the EXTIx software interrupt / event request

7.6.6. Pending register (EXTI_PD)

Address offset: 0x14

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
Reserved										PD21	PD20	PD19	PD18	PD17	PD16
										rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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: 22	Reserved	Must be kept at reset value.
21: 0	PDx	Interrupt pending status x (x = 0…21) 0: EXTI linex is not triggered 1: EXTI linex is triggered. This bit is cleared to 0 by writing 1 to it.

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

8.1. 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 (EXIT).

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.

8.2. 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.

8.3. 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 8-1. GPIO configuration table* shows the details.

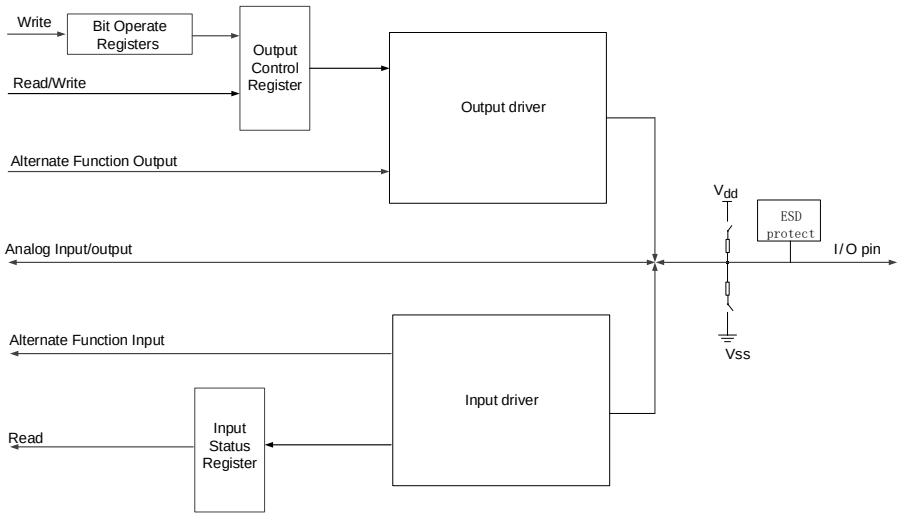
Table 8-1. GPIO configuration table

Configuration mode		CTL[1:0]	SPDy: MD[1:0]	OCTL
Input	Analog	00	x 00	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	x 00: Reserved x 01: Speed up to 10MHz x 10: Speed up to 2MHz 0 11: Speed up to 50MHz 1 11: Speed up to 120MHz(1) (SPDy required to be set to 0b1)	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

When the port output speed is more than 50 MHz, the user should enable the I/O compensation cell. Refer to IO compensation control register (AFIO_CPSCTL).

Figure 8-1. Basic structure of a general-pupose I/O shows the basic structure of an I/O port bit.

Figure 8-1. Basic structure of a general-pupose I/O



8.3.1. 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.

8.3.2. External interrupt/event lines

The port can use external interrupt/event lines only if it is configured in input mode.

8.3.3. 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.

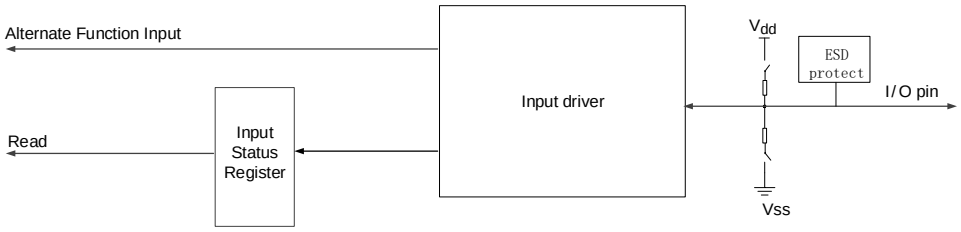
8.3.4. Input configuration

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.
- Disable the output buffer.

Figure 8-2. Basic structure of Input configuration shows the input configuration.

Figure 8-2. Basic structure of Input configuration



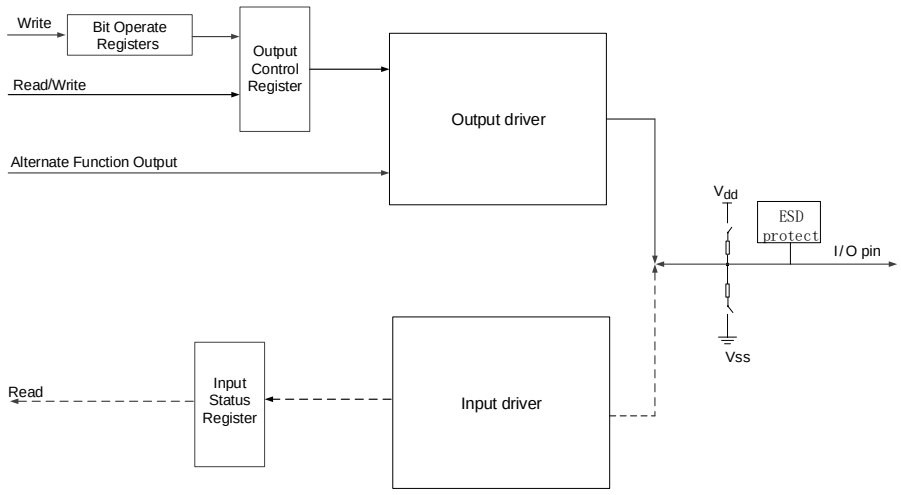
8.3.5. Output configuration

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 8-3. Basic structure of Output configuration shows the output configuration.

Figure 8-3. Basic structure of Output configuration



8.3.6. 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 8-4. Basic structure of Analog configuration shows the analog configuration.

Figure 8-4. Basic structure of Analog configuration

8.3.7. Alternate function (AF) configuration

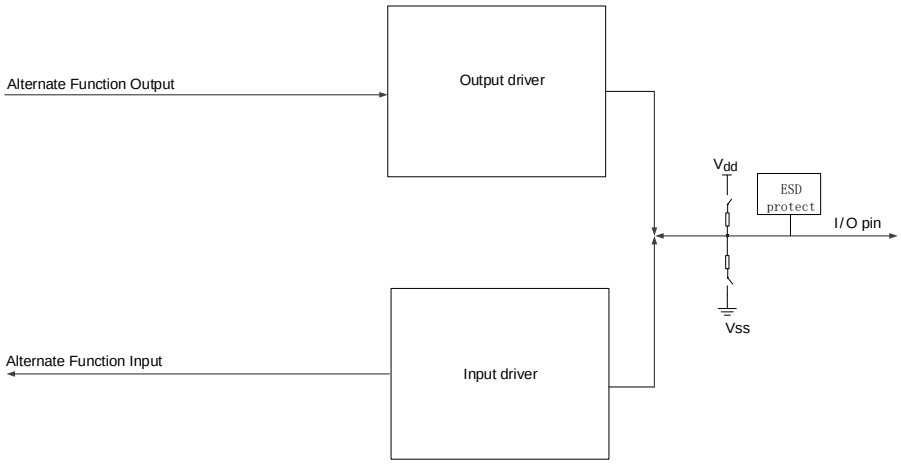
To suit for different device packages, the GPIO supports some alternate functions mapped to some other pins by software.

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 8-5. Basic structure of Alternate function configuration shows the alternate function configuration.

Figure 8-5. Basic structure of Alternate function configuration



8.3.8. 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).

Some alternate function need to configure, the configuration registers are AFIO_PCFA-AFIO_PCFE, and AFIO_PCFG.

8.3.9. 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.

8.3.10. GPIO I/O compensation cell

By default, the I/O compensation cell is not used. However, when the I/O port output speed is more than 50MHz, it is recommended to use the compensation cell for slew rate control to reduce the I/O noise on power supply.

When the compensation cell is enabled, a complete flag CPS_RDY is set to indicate that the compensation cell is ready and can be used.

8.4. Remapping function I/O and debug configuration

8.4.1. 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.

8.4.2. Main features

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

8.4.3. JTAG/SWD alternate function remapping

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

Table 8-2. Debug interface signals

Pin Name	Function description
PA13	JTMS / SWDIO
PA14	JTCK / SWCLK
PA15	JTDI
PB3	JTDO / TRACESWO
PB4	NJTRST

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 8-3. Debug port mapping and Pin availability

SWJ_FCFG [2:0]	JTAG-DP and SW-DP	Pin availability				
		PA13	PA14	PA15	PB3	PB4
000	JTAG-DP Enabled and SW-DP Enabled (Reset state)	X	X	X	X	X
001	JTAG-DP Enabled and SW-DP Enabled but without NJTRST	X	X	X	X	√
010	JTAG-DP Disabled and SW-DP Enabled	X	X	√	√	√
100	JTAG-DP Disabled and SW-DP Disabled	√	√	√	√	√
Other	Forbidden					

1. Can't released if using asynchronous trace.
2. " √ " Indicates that the corresponding pin can be used as a general-purpose I/O.
3. "X" Indicates that the corresponding pin can't be used as a general-purpose I/O.
4. 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

8.4.4. ADC AF remapping

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

Table 8-4. ADC0/1 external trigger routine conversion AF remapping function (1)

Register	ADC0	ADC1
ADC0_ETRGRT_REMAP = 0	ADC0 externa signal trigger routine conversion is connected to EXTI11	-
ADC0_ETRGRT_REMAP	ADC0 external signal trigger	-

Register	ADC0	ADC1
= 1	routine conversion is connected to TIMER7_TRGO	
ADC1_ETRGRT_REMAP = 0	-	ADC1 external signal trigger routine conversion is connected to EXTI11
ADC1_ETRGRT_REMAP = 1	-	ADC1 external signal trigger routine conversion is connected to TIMER7_TRGO

1. Remap available only for High-density devices

8.4.5. TIMER AF remapping

Table 8-5. TIMERx alternate function remapping

Alternate function	TIMERx_REMAP [1:0](x = 0，1，2)			
	TIMERx_REMAP(x = 8，9，10，12，13)		-	
	“0” /“00” (no remap)	“1” /“01” (partial remap)	“10” (partial remap)	“11” (full remap)
TIMER0_ETI	PA12		-	PE7
TIMER0_CH0	PA8		-	PE9
TIMER0_CH1	PA9		-	PE11
TIMER0_CH2	PA10		-	PE13
TIMER0_CH3	PA11		-	PE14
TIMER0_BRKIN	PB12	PA6	-	PE15
TIMER0_CH0_ON	PB13	PA7	-	PE8
TIMER0_CH1_ON	PB14	PB0	-	PE10
TIMER0_CH2_ON	PB15	PB1	-	PE12
TIMER1_CH0/TIMER1_ETI (2)	PA0	PA15	PA0	PA15
TIMER1_CH1	PA1	PB3	PA1	PB3
TIMER1_CH2	PA2		PB10	
TIMER1_CH3	PA3		PB11	
TIMER2_CH0	PA6	-	PB4	PC6
TIMER2_CH1	PA7	-	PB5	PC7

Alternate function	TIMERx_REMAP [1:0](x = 0，1，2)			
	TIMERx_REMAP(x = 8，9，10，12，13)		-	
	“0” /“00” (no remap)	“1” /“01” (partial remap)	“10” (partial remap)	“11” (full remap)
TIMER2_CH2	PB0	-	PB0	PC8
TIMER2_CH3	PB1	-	PB1	PC9
TIMER3_CH0	PB6	PD12	-	-
TIMER3_CH1	PB7	PD13	-	-
TIMER3_CH2	PB8	PD14	-	-
TIMER3_CH3	PB9	PD15	-	-
TIMER8_CH0	PA2	PE5	-	-
TIMER8_CH1	PA3	PE6	-	-
TIMER9_CH0	PB8	PF6	-	-
TIMER10_CH0	PB9	PF7	-	-
TIMER12_CH0	PA6	PF8	-	-
TIMER13_CH0	PA7	PF9	-	-

1. TIMER0 remap(full remap) available only for 100-pin and 144-pin packages
2. TIMER1_CH0 and TIMER1_ETI share the same pin but cannot be used at the same time
3. TIMER2 remap(full remap) available only for 64-pin, 100-pin and 144-pin packages.
4. TIMER3 remap(full remap) available only for 100-pin and 144-pin packages.
5. TIMER8/9/10/12/13 refer to the AF remap and debug I/O configuration register 1(AFIO_PCF1).

Table 8-6. TIMER4 alternate function remapping (1)

Alternate function	TIMER4CH3_IREMAP = 0	TIMER4CH3_IREMAP = 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 and Connectivity lines devices.

8.4.6. USART AF remapping

Refer to AFIO port configuration register 0 (AFIO_PCF0).

Table 8-7. USART0/1/2 alternate function remapping

Register	USART0	USART1	USART2
USART0_REMAP = 0	PA9(USART0_TX) PA10(USART0_RX)		-

USART0_REMAP = 1	PB6(USART0_TX) PB7(USART0_RX)		-
USART1_REMAP = 0	-	PA0(USART1_CTS) PA1(USART1_RTS) PA2(USART1_TX) PA3(USART1_RX) PA4(USART1_CK)	-
USART1_REMAP = 1 (1)	-	PD3(USART1_CTS) PD4(USART1_RTS) PD5(USART1_TX) PD6(USART1_RX) PD7(USART1_CK)	-
USART2_REMAP[1:0] = “00” (no remap)	-	-	PB10(USART2_TX) PB11(USART2_RX) PB12(USART2_CK) PB13(USART2_CTS) PB14(USART2_RTS)
USART2_REMAP [1:0] =“01” (partial remap) (2)	-	-	PC10(USART2_TX) PC11(USART2_RX) PC12(USART2_CK) PB13(USART2_CTS) PB14(USART2_RTS)
USART2_REMAP [1:0] =“11” (full remap) (3)	-	-	PD8(USART2_TX) PD9(USART2_RX) PD10(USART2_CK) PD11(USART2_CTS) PD12(USART2_RTS)

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

8.4.7. I2C0 AF remapping

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

Table 8-8. I2C0 alternate function remapping

Register	I2C0_SCL	I2C0_SDA
I2C0_REMAP = 0	PB6	PB7
I2C0_REMAP = 1	PB8	PB9

8.4.8. SPI0/SPI2/I2S AF remapping

Refer to AFIO port configuration register 0 (AFIO_PCF0).

Table 8-9. SPI0/SPI2/I2S alternate function remapping

Register	SPI0	SPI2/I2S
SPI0_REMAP = 0	PA4(SPI0_NSS) PA5(SPI0_SCK) PA6(SPI0_MISO) PA7(SPI0_MOSI)	-
SPI0_REMAP = 1	PA15(SPI0_NSS) PB3(SPI0_SCK) PB4(SPI0_MISO) PB5(SPI0_MOSI)	-
SPI2_REMAP = 0	-	PA15(SPI2_NSS/ I2S2_WS) PB3(SPI2_SCK/ I2S2_CK) PB4(SPI2_MISO) PB5(SPI2_MOSI/I2S2_SD)
SPI2_REMAP = 1	-	PA4(SPI2_NSS/ I2S2_WS) PC10(SPI2_SCK/ I2S2_CK) PC11(SPI2_MISO) PC12(SPI2_MOSI/I2S2_SD)

8.4.9. CAN0/1 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 possible only in devices delivered in 100-pin and 144-pin packages.

Table 8-10. CAN0/1 alternate function remapping

Register (1)	CAN0	CAN1
CAN0_REMAP[1:0] =“00”	PA11(CAN0_RX) PA12(CAN0_TX)	-
CAN0_REMAP[1:0] =“10”	PB8(CAN0_RX) PB9(CAN0_TX)	-
CAN0_REMAP[1:0] =“11”	PD0(CAN0_RX) PD1(CAN0_TX)	-
CAN1_REMAP = “0”	-	PB12(CAN1_RX) PB13(CAN1_TX)
CAN1_REMAP = “1”	-	PB5(CAN1_RX) PB6(CAN1_TX)

1. CAN0_RX and CAN0_TX in connectivity line devices; CAN_RX and CAN_TX in other devices with a single CAN interface.

8.4.10. Ethernet AF remapping

Table 8-11. ENET alternate function remapping

Register	ENET
ENET_REMAP = “0”	PA7(RX_DV-CRS_DV) PC4(RXD0) PC5(RXD1) PB0(RXD2) PB1(RXD3)
ENET_REMAP = “1”	PD8(RX_DV-CRS_DV) PD9(RXD0) PD10(RXD1) PD11(RXD2) PD12(RXD3)

8.4.11. CTC AF remapping

Refer to AFIO port configuration register 1 (AFIO_ PCF1).

Table 8-12. CTC alternate function remapping

Alternate function	CTC_REMAP [1:0] = “00”	CTC_REMAP [1:0] = “01”	CTC_REMAP [1:0] = “10” or “11”
CTC_SYNC	PA8	PD15	PF0

8.4.12. 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 VBAT (VDD 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 *4.3.1*.

Table 8-13. 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 8-14. OSC pins configuration

Alternate function	HXTAL= ON	HXTAL = OFF
PD0	OSC_IN	PD0

PD1	OSC_OUT	PD1

8.5. Register definition

GPIOA base address: 0x4001 0800
GPIOB base address: 0x4001 0C00
GPIOC base address: 0x4001 1000
GPIOD base address: 0x4001 1400
GPIOE base address: 0x4001 1800
GPIOF base address: 0x4001 1C00
GPIOG base address: 0x4001 2000
AFIO base address: 0x4001 0000

8.5.1. 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]	Port 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(10MHz) 10: Output mode(2MHz) 11: Output mode(50MHz)

8.5.2. 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	

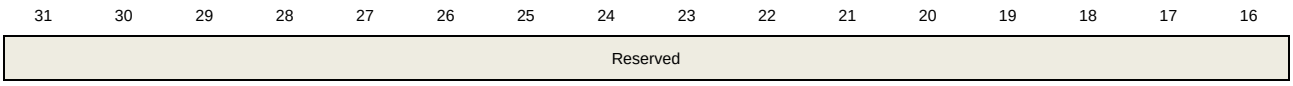
Bits	Fields	Descriptions
31:30	CTL15[1:0]	Port 15 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
29:28	MD15[1:0]	Port 15 mode bits These bits are set and cleared by software refer to MD0[1:0]description
27:26	CTL14[1:0]	Port 14 configuration bits These bits 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

8.5.3. 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).



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

8.5.4. 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

8.5.5. 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

8.5.6. 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

8.5.7. 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

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”

8.5.8. Port bit speed register (GPIOx_ SPD, x=A..G)

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
SPD15	SPD14	SPD13	SPD12	SPD11	SPD10	SPD9	SPD8	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

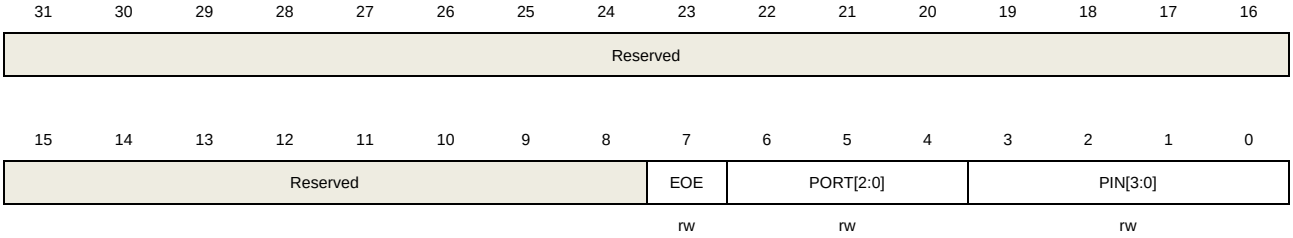
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	SPDy	Set very high output speed(120MHz) when MDx is 0b11. If the port output speed is more than 50MHz, set this bit to 1 and set MDx to 0b11.These bits are set and cleared by software. 0: No effect 1: Max speed more than 50MHz.(MDx required to be set to 0b11 together) Note: When the port output speed is more than 50 MHz, the user should enable the I/O compensation cell. Refer to CPS_EN bit in AFIO_CPSCTL register.

8.5.9. Event control register (AFIO_EC)

Address offset: 0x00

Reset value: 0x0000 0000

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



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

8.5.10. 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 High-density devices:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			SPI2_REMAP	Reserved	SWJ_CFG[2:0]			Reserved	CAN1_REMAP	Reserved	ADC1_ETRGRT_REMAP	Reserved	ADC0_ETRGRT_REMAP	Reserved	TIMER4CH3_IREMAP
			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_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 reset by software. 0: Disable the remapping function (SPI2_NSS-I2S2_WS/PA15, SPI2_SCK-I2S2_CK/PB3, SPI2_MISO/PB4, SPI2_MOSI-I2S_SD/PB5) 1: Enable the remapping function fully (SPI2_NSS-I2S2_WS/PA4, SPI2_SCK-I2S2_CK/PC10, SPI2_MISO/PC11, SPI2_MOSI-I2S_SD/PC12)
27	Reserved	Must be kept at reset value
26:24	SWJ_CFG[2:0]	Serial wire JTAG configuration These bits are write-only. 000: JTAG-DP Enabled and SW-DP Enabled(Reset State) 001: JTAG-DP Enabled and SW-DP Enabled , but without NJTRST 010: Disable JTAG-DP and Enable SW-DP 100: Disable JTAG-DP and SW-DP Other: Undefined
23	Reserved	Must be kept at reset value.
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: Disable the remapping function (CAN1_RX/PB12,CAN_TX/PB13) 1: Enable the remapping function (CAN1_RX/PB5,CAN_TX/PB6).
21	Reserved	Must be kept at reset value.
20	ADC1_ETRGRT_REMAP	ADC 1 external trigger routine conversion remapping This bit is set and reset by software. 0: Connect the ADC1 external signal trigger routine conversion to EXTI11. 1: Connect the ADC1 external signal trigger routine conversion to TIM7_TRGO.
19	Reserved	Must be kept at reset value.

18	ADC0_ETRGRT_REMAP	ADC 0 external trigger routine conversion remapping This bit is set and reset by software. 0: Connect the ADC0 external signal trigger routine conversion to EXTI11. 1: Connect the ADC0 external signal trigger routine conversion to TIM7_TRGO.
17	Reserved	Must be kept at reset value.
16	TIMER4CH3_IREMAP	TIMER4 channel3 internal remapping This bit is set and reset by software. 0: ConnectTIMER4_CH3 to PA3. 1: Connect the IRC40K internal clock to TIMER4_CH3 input in order to calibration.
15	PD01_REMAP	Port D0/Port D1 mapping to OSC_IN/OSC_OUT This bit is set and cleared by software. 0: Not remap 1: PD0 remapped to OSC_IN, PD1 remapped to OSC_OUT
14:13	CAN0_REMAP [1:0]	CAN0 alternate interface remapping These bits are set and cleared by software. 00: Disable the remapping function (CAN0_RX/PA11,CAN0_TX/PA12) 01: Not used 10: Enable the remapping function partially(CAN0_RX/PB8,CAN0_TX/PB9) 11: Enable the remapping function fully (CAN0_RX/PD0,CAN0_TX/PD1)
12	TIMER3_REMAP	TIMER3 remapping This bit is set and reset by software 0: Disable the remapping function (TIMER3_CH0/PB6, TIMER3_CH1/PB7, TIMER3_CH2/PB8, TIMER3_CH3/PB9) 1: Enable the remapping function fully(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 reset by software 00: Disable the remapping function(TIMER2_CH0/PA6, TIMER2_CH1/PA7, TIMER2_CH2/PB0, TIMER2_CH3/PB1) 01: Not used 10: Enable the remapping function partially (TIMER2_CH0/PB4, TIMER2_CH1/PB5, TIMER2_CH2/PB0, TIMER2_CH3/PB1) 11: Enable the remapping function fully (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 reset by software 00: Disable the remapping function(TIMER1_CH0-TIMER1_ETI/PA0, TIMER1_CH1/PA1, TIMER1_CH2/PA2, TIMER1_CH3/PA3) 01: Enable the remapping function partially(TIMER1_CH0-TIMER1_ETI/PA15, TIMER1_CH1/PB3, TIMER1_CH2/PA2, TIMER1_CH3/PA3)

		10: not used 11: Enable the remapping function fully(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 reset by software 00: Disable the remapping function(TIMER0_ETI / PA12, TIMER0_CH0 / PA8, TIMER0_CH1 / PA9, TIMER0_CH2 / PA10, TIMER0_CH3 / PA11, TIMER0_BRKIN / PB12, TIMER0_CH0_ON / PB13, TIMER0_CH1_ON / PB14, TIMER0_CH2_ON / PB15) 01: Enable the remapping function partially(TIMER0_ETI / PA12, TIMER0_CH0 / PA8, TIMER0_CH1 / PA9, TIMER0_CH2 / PA10, TIMER0_CH3 / PA11, TIMER0_BRKIN / PA6, TIMER0_CH0_ON / PA7, TIMER0_CH1_ON / PB0, TIMER0_CH2_ON / PB1) 10: Not used 11: Enable the remapping function fully(TIMER0_ETI / PE7, TIMER0_CH0 / PE9, TIMER0_CH1 / PE11, TIMER0_CH2 / PE13, TIMER0_CH3 / PE14, TIMER0_BRKIN / 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 reset by software 00: Disable the remapping function (USART2_TX/PB10, USART2_RX /PB11, USART2_CK/PB12, USART2_CTS/PB13, USART2_RTS/PB14) 01: Enable the remapping function partially(USART2_TX/PC10, USART2_RX /PC11, USART2_CK/PC12, USART2_CTS/PB13, USART2_RTS/PB14) 10: Not used 11: Enable the remapping function fully(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 reset by software 0: Disable the remapping function(USART1_CTS/PA0, USART1_RTS/PA1, USART1_TX/PA2, USART1_RX /PA3, USART1_CK/PA4) 1: Enable the remapping function(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 reset by software 0: Disable the remapping function (USART0_TX/PA9, USART0_RX /PA10) 1: Enable the remapping function (USART0_TX/PB6, USART0_RX /PB7)
1	I2C0_REMAP	I2C0 remapping This bit is set and reset by software 0: Disable the remapping function (I2C0_SCL/PB6, I2C0_SDA /PB7) 1: Enable the remapping function (I2C0_SCL/PB8, I2C0_SDA /PB9)

0	SPI0_REMAP	SPI0 remapping This bit is set and cleared by software. 0: Disable the remapping function (SPI0_NSS/PA4, SPI0_SCK /PA5, SPI0_MISO /PA6,SPI0_MOSI /PA7) 1: Enable the remapping function (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 0: PB5 pin is used as a general-purpose I/O 1: Enale PPT_PPS to be output on PB5
29	TIMER1ITI1_REMAP	TIMER1 internal trigger 1 remapping It control the TMER1_ITI1 mapping 0: TMER1_ITI1 is used as the Ethernet PTP output in order to calibration. 1: TMER1_ITI1 is used as the USB OTG SOF (Start of Frame) output in order to calibration.
28	SPI2_REMAP	SPI2/I2S2 remapping 0: Disable the remapping function (SPI2_NSS-I2S2_WS/PA15, SPI2_SCK-I2S2_CK/PB3, SPI2_MISO/PB4, SPI2_MOSI-I2S_SD/PB5) 1: Enable the remapping function fully (SPI2_NSS-I2S2_WS/PA4, SPI2_SCK-I2S2_CK/PC10, SPI2_MISO/PC11, SPI2_MOSI-I2S_SD/PC12)
27	Reserved	Must be kept at reset value
26:24	SWJ_CFG[2:0]	Serial wire JTAG configuration Software can only write to this bit. 000: JTAG-DP Enabled and SW-DP Enabled(Reset State) 001: JTAG-DP Enabled and SW-DP Enabled , but without NJTRST 010: Disable JTAG-DP and Enable SW-DP 100: Disable JTAG-DP and SW-DP

		Other: Undefined
23	ENET_PHY_SEL	Ethernet MII or RMII PHY selection 0:Select an MII PHY 1: Select an RMII PHY
22	CAN1_REMAP	CAN1 I/O remapping 0: Disable the remapping function (CAN1_RX/PB12,CAN_TX/PB13) 1: Enable the remapping function (CAN1_RX/PB5,CAN_TX/PB6)
21	ENET_REMAP	Ethernet MAC I/O remapping 0: Disable the remapping function (RX_DV-CRS_DV/PA7, RXD0/PC4, RXD1/PC5, RXD2/PB0, RXD3/PB1) 1: Enable the remapping function (RX_DV-CRS_DV/PD8, RXD0/PD9, RXD1/PD10, RXD2/PD11, RXD3/PD12)
20:17	Reserved	Must be kept at reset value
16	TIMER4CH3_IREMAP	TIMER4 channel3 internal remapping 0: Connect TIMER4_CH3 to PA3. 1:Connect the IRC40K internal clock to TIMER4_CH3 input in order to calibration.
15	PD01_REMAP	Port D0/D1 mapping on OSC_IN/OSC_OUT 0: Disable the remapping function 1: PD0 is used as OSC_IN, PD1 is used as OSC_OUT
14:13	CAN0_REMAP[1:0]	CAN0 alternate interface remapping 00: Disable the remapping function (CAN0_RX/PA11,CAN0_TX/PA12) 01: Not used 10: Enable the remapping function partially (CAN0_RX/PB8,CAN0_TX/PB9) 11: Enable the remapping function fully (CAN0_RX/PD0,CAN0_TX/PD1)
12	TIMER3_REMAP	TIMER3 remapping 0: Disable the remapping function (TIMER3_CH0/PB6, TIMER3_CH1/PB7, TIMER3_CH2/PB8, TIMER3_CH3/PB9) 1: Enable the remapping function fully (TIMER3_CH0/PD12, TIMER3_CH1/PD13, TIMER3_CH2/PD14, TIMER3_CH3/PD15)
11:10	TIMER2_REMAP[1:0]	TIMER2 remapping 00: Disable the remapping function (TIMER2_CH0/PA6, TIMER2_CH1/PA7, TIMER2_CH2/PB0, TIMER2_CH3/PB1) 01: Not used 10: Enable the remapping function partially (TIMER2_CH0/PB4,TIMER2_CH1/PB5, TIMER2_CH2/PB0,TIMER2_CH3/PB1) 11: Enable the remapping function fully (TIMER2_CH0/PC6, TIMER2_CH1/PC7, TIMER2_CH2/PC8, TIMER2_CH3/PC9)
9:8	TIMER1_REMAP[1:0]	TIMER1 remapping 00: Disable the remapping function (TIMER1_CH0-TIMER1_ETI/PA0,

		TIMER1_CH1/PA1, TIMER1_CH2/PA2, TIMER1_CH3/PA3) 01: Enable the remapping function partially 0 (TIMER1_CH0-TIMER1_ETI/PA15, TIMER1_CH1/PB3, TIMER1_CH2/PA2, TIMER1_CH3/PA3) 10: Enable the remapping function partially 1 (TIMER1_CH0-TIMER1_ETI/PA0, TIMER1_CH1/PA1, TIMER1_CH2/PB10, TIMER1_CH3/PB11) 11: Enable the remapping function fully (TIMER1_CH0-TIMER1_ETI/PA15, TIMER1_CH1/PB3, TIMER1_CH2/PB10, TIMER1_CH3/PB11)
7:6	TIMER0_REMAP[1:0]	TIMER0 remapping 00: Disable the remapping function (TIMER0_ETI / PA12, TIMER0_CH0 / PA8, TIMER0_CH1 / PA9, TIMER0_CH2 / PA10, TIMER0_CH3 / PA11, TIMER0_BRKIN / PB12, TIMER0_CH0_ON / PB13, TIMER0_CH1_ON / PB14, TIMER0_CH2_ON / PB15) 01: Enable the remapping function partially (TIMER0_ETI / PA12, TIMER0_CH0 / PA8, TIMER0_CH1 / PA9, TIMER0_CH2 / PA10, TIMER0_CH3 / PA11, TIMER0_BRKIN / PA6, TIMER0_CH0_ON / PA7, TIMER0_CH1_ON / PB0, TIMER0_CH2_ON / PB1) 10: Not used 11: Enable the remapping function fully (TIMER0_ETI/PE7, TIMER0_CH0/ PE9, TIMER0_CH1/PE11,TIMER0_CH2/PE13,TIMER0_CH3/PE14, TIMER0_BRKIN/PE15, TIMER0_CH0_ON/PE8, TIMER0_CH1_ON/PE10, TIMER0_CH2_ON/PE12)
5:4	USART2_REMAP[1:0]	USART2 remapping 00: Disable the remapping function (USART2_TX/PB10, USART2_RX /PB11, USART2_CK/PB12,USART2_CTS/PB13, USART2_RTS/PB14) 01: Enable the remapping function partially (USART2_TX/PC10, USART2_RX /PC11, USART2_CK/PC12,USART2_CTS/PB13, USART2_RTS/PB14) 10: Not used 11: Enable the remapping function fully (USART2_TX/PD8, USART2_RX /PD9, USART2_CK/PD10,USART2_CTS/PD11, USART2_RTS/PD12)
3	USART1_REMAP	USART1 remapping 0: Disable the remapping function (USART1_CTS/PA0, USART1_RTS/PA1, USART1_TX/PA2, USART1_RX /PA3, USART1_CK/PA4) 1: Enable the remapping function (USART1_CTS/PD3, USART1_RTS/PD4, USART1_TX/PD5, USART1_RX /PD6, USART1_CK/PD7)
2	USART0_REMAP	USART0 remapping 0: Disable the remapping function (USART0_TX/PA9, USART0_RX /PA10) 1: Enable the remapping function (USART0_TX/PB6, USART0_RX /PB7)
1	I2C0_REMAP	I2C0 remapping 0: Disable the remapping function (I2C0_SCL/PB6, I2C0_SDA /PB7) 1: Enable the remapping function (I2C0_SCL/PB8, I2C0_SDA /PB9)
0	SPI0_REMAP	SPI0 remapping

0: Disable the remapping function (SPI0_NSS/PA4, SPI0_SCK /PA5, SPI0_MISO /PA6, SPI0_MOSI /PA7)

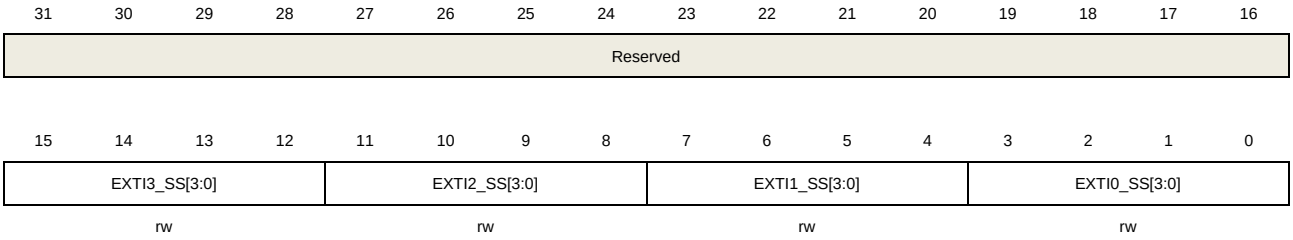
1: Enable the remapping function (SPI0_NSS/PA15, SPI0_SCK /PB3, SPI0_MISO /PB4, SPI0_MOSI /PB5)

8.5.11. EXTI sources selection register 0 (AFIO_EXTISS0)

Address offset: 0x08

Reset value: 0x0000 0000

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



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.

8.5.12. 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.

8.5.13. 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.

8.5.14. 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

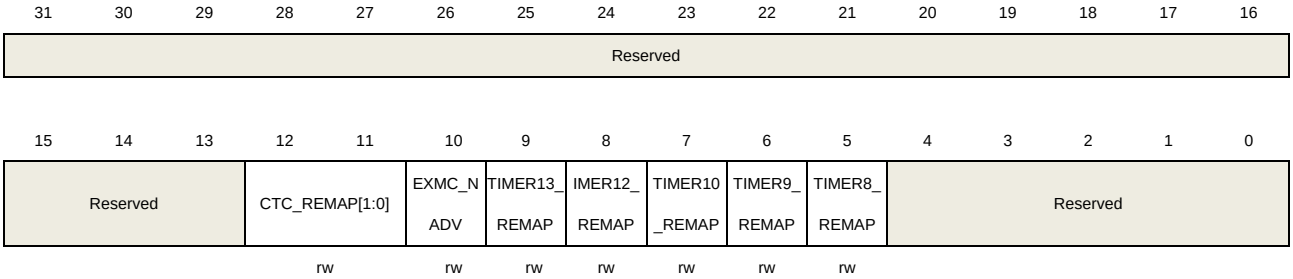
Other configurations are reserved.

8.5.15. AFIO port configuration register 1 (AFIO_PCF1)

Address offset: 0x1C

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value
12:11	CTC_REMAP[1:0]	CTC remapping These bits are set and cleared by software, it controls the mapping of the CTC_SYNC alternate function onto the GPIO ports 00: Disable the remapping function (PA8) 01: Enable the remapping function 0 (PD15) 10/11: Enable the remapping function 1 (PF0)
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: Disable the remapping function (PA7) 1: Enable the remapping function (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: Disable the remapping function (PA6) 1: Enable the remapping function (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: Disable the remapping function (PB9) 1: Enable the remapping function (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: Disable the remapping function (PB8) 1: Enable the remapping function (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: Disable the remapping function (TIMER8_CH0 on PA2 and TIMER8_CH1 on PA3) 1: Enable the remapping function (PF6) (TIMER8_CH0 on PE5 and TIMER8_CH1 on PE6)
4:0	Reserved	Must be kept at reset value

8.5.16. IO compensation control register (AFIO_CPSCTL)

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							CPS_RDY	Reserved							CPS_EN
							r								rw

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	CPS_RDY	I/O compensation cell is ready or not. This bit is read-only. 0: I/O compensation cell is not ready

		1: I/O compensation cell is ready
7:1	Reserved	Must be kept at reset value.
0	CPS_EN	I/O compensation cell enable. When the port output speed is more than 50 MHz, the user should enable the I/O compensation cell. 0: I/O compensation cell is power-down 1: I/O compensation cell is enable

8.5.17. AFIO port configuration register A (AFIO_PCFA)

Address offset: 0x3C

Reset value: 0x0000 0000

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

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30	PA15_AFCFG	PA15 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PA15 alternate function to SHRTIMER 1: Configure PA15 alternate function to SHRTIMER
29:26	Reserved	Must be kept at reset value.
25:24	PA12_AFCFG[1:0]	PA12 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PA12 alternate function to SHRTIMER/ CMP1/USART5 01: Configure PA12 alternate function select to enable CMP1 10: Configure PA12 alternate function select to enable USART5 11: Configure PA12 alternate function select to enable SHRTIMER
23:22	PA11_AFCFG[1:0]	PA11 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PA11 alternate function to enable SHRTIMER/USART5 01: Configure PA11 alternate function to USART5

		10/11: Configure PA11 alternate function to SHRTIMER
21:20	PA10_AFCFG[1:0]	PA10 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PA10 alternate function to SHRTIMER/CAN2/CMP5 01: Configure PA10 alternate function to CAN2 10: Configure PA10 alternate function to CMP5 11: Configure PA10 alternate function to SHRTIMER **Note**: CAN2 is available only in connectivity line devices.
19:18	PA9_AFCFG[1:0]	PA9 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PA9 alternate function to SHRTIMER/CAN2/I2C2 01: Configure PA9 alternate function to CAN2 10: Configure PA9 alternate function to I2C2 11: Configure PA9 alternate function to SHRTIMER **Note**: CAN2 is available only in connectivity line devices.
17:16	PA8_AFCFG[1:0]	PA8 AF function configuration bits These bits are set and cleared by software. 00: Donot configure PA8 alternate function to SHRTIMER/I2C2 01: Configure PA8 alternate function to I2C2 10/11: Configure PA8 alternate function to SHRTIMER
15:11	Reserved	Must be kept at reset value.
10	PA5_AFCFG	PA5 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PA5 alternate function to USBHS 1: Configure PA5 alternate function to USBHS
9:7	Reserved	Must be kept at reset value.
6	PA3_AFCFG	PA3 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PA3 alternate function to USBHS 1: Configure PA3 alternate function to USBHS
5	Reserved	Must be kept at reset value.
4	PA2_AFCFG	PA2 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PA2 alternate function to CMP1 1: Configure PA2 alternate function to CMP1
3:0	Reserved	Must be kept at reset value.

8.5.18. AFIO port configuration register B (AFIO_PCFB)

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	PB15_AFCFG	PB14_AFCFG[1:0]		PB13_AFCFG[1:0]		PB12_AFCFG[1:0]		PB11_AFCFG[1:0]		PB10_AFCFG[1:0]		PB9_AFCFG[1:0]		PB8_AFCFG[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
Reserved	PB7_AFCFG	Reserved	PB6_AFCFG	PB5_AFCFG[1:0]		PB4_AFCFG[1:0]		Reserved	PB3_AFCFG	PB2_AFCFG[1:0]		PB1_AFCFG[1:0]		Reserved	PB0_AFCFG
	rw		rw	rw		rw			rw	rw		rw			rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30	PB15_AFCFG	PB15 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PB15 alternate function to SHRTIMER 1: Configure PB15 alternate function to SHRTIMER
29:28	PB14_AFCFG[1:0]	PB14 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB14 alternate function to SHRTIMER/I2S1 01: Configure PB14 alternate function to I2S1 10/11: Configure PB14 alternate function to SHRTIMER
27:26	PB13_AFCFG[1:0]	PB13 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB13 alternate function to SHRTIMER/USBHS 01/11: Configure PB13 alternate function to SHRTIMER 10: Configure PB13 alternate function to USBHS
25:24	PB12_AFCFG[1:0]	PB12 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB12 alternate function to SHRTIMER/USBHS 01/11: Configure PB12 alternate function to SHRTIMER 10: Configure PB12 alternate function to USBHS
23:22	PB11_AFCFG[1:0]	PB11 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB11 alternate function to SHRTIMER/USBHS/CAN2 01: Configure PB11 alternate function to CAN2 10: Configure PB11 alternate function to USBHS

		11: Configure PB11 alternate function to SHRTIMER **Note**: CAN2 is available only in connectivity line devices.
21:20	PB10_AFCFG[1:0]	PB10 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB10 alternate function to SHRTIMER/USBHS/CAN2 01: Configure PB10 alternate function to CAN2 10: Configure PB10 alternate function to USBHS 11: Configure PB10 alternate function to SHRTIMER **Note**: CAN2 is available only in connectivity line devices.
19:18	PB9_AFCFG[1:0]	PB9 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB9 alternate function to SHRTIMER/CMP1 01: Configure PB9 alternate function to CMP1 10/11: Configure PB9 alternate function to SHRTIMER
17:16	PB8_AFCFG[1:0]	PB8 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB8 alternate function to SHRTIMER/I2C2 01: Configure PB8 Alternate function to I2C2 10/11: Configure PB8 alternate function to SHRTIMER
15	Reserved	Must be kept at reset value
14	PB7_AFCFG	PB7 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PB7 alternate function to SHRTIMER 1: Configure PB7 alternate function to SHRTIMER
13	Reserved	Must be kept at reset value
12	PB6_AFCFG	PB6 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PB6 alternate function to SHRTIMER 1: Configure PB6 alternate function to SHRTIMER
11:10	PB5_AFCFG[1:0]	PB5 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB5 alternate function to SHRTIMER/USBHS/I2C2 01: Configure PB5 alternate function to I2C2 10: Configure PB5 alternate function to USBHS 11: Configure PB5 alternate function to SHRTIMER
9:8	PB4_AFCFG[1:0]	PB4 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB4 alternate function to SHRTIMER/I2C2/I2S2 01: Configure PB4 alternate function to I2S2

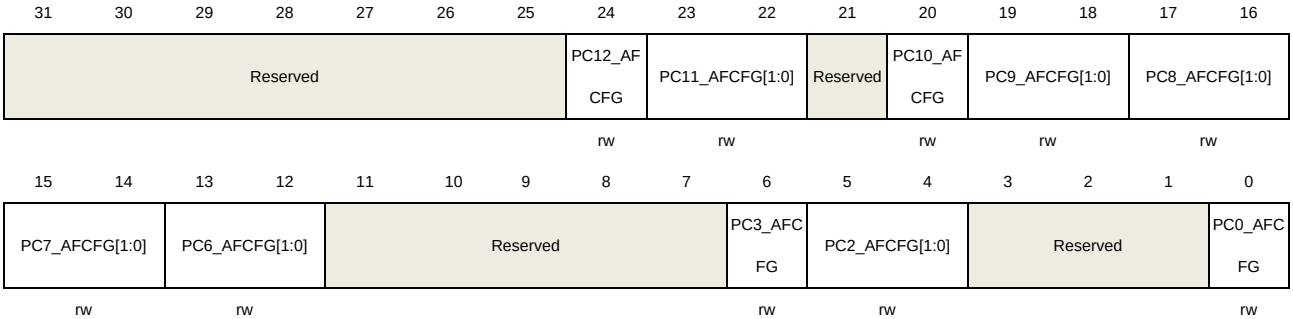
		10: Configure PB4 alternate function to I2C2 11: Configure PB4 alternate function to SHRTIMER
7	Reserved	Must be kept at reset value
6	PB3_AFCFG	PB3 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PB3 alternate function to SHRTIMER 1: Configure PB3 alternate function to SHRTIMER
5:4	PB2_AFCFG[1:0]	PB2 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB2 alternate function to SHRTIMER/USBHS 10: Configure PB2 alternate function to USBHS 01/11: Configure PB2 alternate function to SHRTIMER
3:2	PB1_AFCFG[1:0]	PB1 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PB1 alternate function to SHRTIMER/USBHS/CMP3 01: Configure PB1 alternate function to CMP3 10: Configure PB1 alternate function to USBHS 11: Configure PB1 alternate function to SHRTIMER
1	Reserved	Must be kept at reset value
0	PB0_AFCFG	PB0 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PB0 alternate function to USBHS 1: Configure PB0 alternate function to USBHS

8.5.19. AFIO port configuration register C (AFIO_PCFC)

Address offset: 0x44

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions

31:25	Reserved	Must be kept at reset value
24	PC12_AFCFG	PC12 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PC12 alternate function to SHRTIMER 1: Configure PC12 alternate function to SHRTIMER
23:22	PC11_AFCFG[1:0]	PC11 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PC11 alternate to SHRTIMER/I2S2 01/11: Configure PC11 alternate function to SHRTIMER 10: Configure PC11 alternate function to I2S2
21	Reserved	Must be kept at reset value
20	PC10_AFCFG	PC10 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PC10 alternate function to I2C2 1: Configure PC10 alternate function to I2C2
19:18	PC9_AFCFG[1:0]	PC9 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PC9 alternate function to SHRTIMER/I2C2 01/11: Configure PC9 alternate function to SHRTIMER 10: Configure PC9 alternate function to I2C2
17:16	PC8_AFCFG[1:0]	PC8 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PC8 alternate function to SHRTIMER/ USART5 10: Configure PC8 alternate function to USART5 01/11: Configure PC8 alternate function to SHRTIMER
15:14	PC7_AFCFG[1:0]	PC7 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PC7 alternate function to SHRTIMER/ USART5 01/11: Configure PC7 alternate function to SHRTIMER 10: Configure PC7 alternate function to USART5
13:12	PC6_AFCFG[1:0]	PC6 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PC6 alternate function to SHRTIMER/CMP5/USART5 01: Configure PC6 alternate function to CMP5 10: Configure PC6 alternate function to USART5 11: Configure PC6 alternate function to SHRTIMER
11:7	Reserved	Must be kept at reset value
6	PC3_AFCFG	PC3 AF function configuration bit This bit is set and cleared by software.

		0: Do not configure PC3 alternate function to USBHS 1: Configure PC3 alternate function to USBHS
5:4	PC2_AFCFG[1:0]	PC2 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PC2 alternate function to USBHS/I2S1 01/11: Configure PC2 alternate function to I2S1 10: Configure PC2 alternate function to USBHS
3:1	Reserved	Must be kept at reset value
0	PC0_AFCFG	PC0 AF function configuration register This bit is set and cleared by software 0: Do not configure PC0 alternate function to USBHS 1: Configure PC0 alternate function to USBHS

8.5.20. AFIO port configuration register D (AFIO_PCFD)

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					PD5_AFCFG	Reserved	PD4_AFCFG	Reserved							
					rw		rw								

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10	PD5_AFCFG	PD5 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PD5 alternate function to SHRTIMER 1: Configure PD5 alternate function to SHRTIMER
9	Reserved	Must be kept at reset value
8	PD4_AFCFG	PD4 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PD4 alternate function to SHRTIMER 1: Configure PD4 alternate function to SHRTIMER
7:0	Reserved	Must be kept at reset value

8.5.21. AFIO port configuration register E (AFIO_PCFE)

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					PE13_AFCFG	Reserved	PE12_AFCFG	Reserved	PE11_AFCFG	Reserved	PE10_AFCFG	Reserved	PE9_AFCFG	Reserved	PE8_AFCFG
					rw		rw		rw		rw		rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PE1_AFCFG[1:0]		PE0_AFCFG[1:0]	
												rw		rw	

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value
26	PE13_AFCFG	PE13 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PE13 alternate function to CMP1 1: Configure PE13 alternate function to CMP1
25	Reserved	Must be kept at reset value
24	PE12_AFCFG	PE12 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PE12 alternate function to CMP3 1: Configure PE12 alternate function to CMP3
23	Reserved	Must be kept at reset value
22	PE11_AFCFG	PE11 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PE11 alternate function to CMP5 1: Configure PE11 alternate function to CMP5
21	Reserved	Must be kept at reset value
20	PE10_AFCFG	PE10 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PE10 alternate function to CMP5 1: Configure PE10 alternate function to CMP5
19	Reserved	Must be kept at reset value
18	PE9_AFCFG	PE9 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PE9 alternate function to CMP3

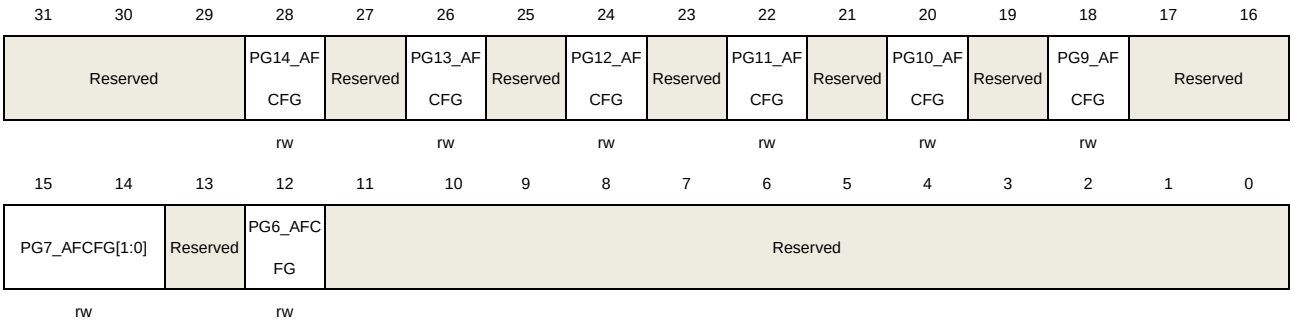
		1: Configure PE9 alternate function to CMP3
17	Reserved	Must be kept at reset value
16	PE8_AFCFG	PE8 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PE8 alternate function to CMP1 1: Configure PE8 alternate function to CMP1
15:4	Reserved	Must be kept at reset value
3:2	PE1_AFCFG[1:0]	PE1 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PE1 alternate function to SHRTIMER/CAN2 01: Configure PE1 alternate function to CAN2 10/11: Configure PE1 alternate function to SHRTIMER **Note**: CAN2 is available only in connectivity line devices.
1:0	PE0_AFCFG[1:0]	PE0 AF function enable configuration bits These bits are set and cleared by software. 00: Do not configure PE0 alternate function to SHRTIMER/CAN2 01: Configure PE0 alternate function to CAN2 10/11: Configure PE0 alternate function to SHRTIMER **Note**: CAN2 is available only in connectivity line devices.

8.5.22. AFIO port configuration register G (AFIO_PCFG)

Address offset: 0x54

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value.
28	PG14_AFCFG	PG14 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PG14 alternate function to USART5

		1: Configure PG14 alternate function to USART5
27	Reserved	Must be kept at reset value.
26	PG13_AFCFG	PG13 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PG13 alternate function to SHRTIMER 1: Configure PG13 alternate function to SHRTIMER
25	Reserved	Must be kept at reset value.
24	PG12_AFCFG	PG12 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PG12 alternate function to SHRTIMER 1: Configure PG12 alternate function to SHRTIMER
23	Reserved	Must be kept at reset value.
22	PG11_AFCFG	PG11 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PG11 alternate function to SHRTIMER 1: Configure PG11 alternate function to SHRTIMER
21	Reserved	Must be kept at reset value.
20	PG10_AFCFG	PG10 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PG10 alternate function to SHRTIMER 1: Configure PG10 alternate function to SHRTIMER
19	Reserved	Must be kept at reset value.
18	PG9_AFCFG	PG9 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PG9 alternate function to USART5 1: Configure PG9 alternate function to USART5
17:16	Reserved	Must be kept at reset value.
15:14	PG7_AFCFG[1:0]	PG7 AF function configuration bits These bits are set and cleared by software. 00: Do not configure PG7 alternate function to SHRTIMER/ USART5 01: Configure PG7 alternate function to USART5 10/11: Configure PG7 alternate function to SHRTIMER
13	Reserved	Must be kept at reset value.
12	PG6_AFCFG	PG6 AF function configuration bit This bit is set and cleared by software. 0: Do not configure PG6 alternate function to SHRTIMER

		1: Configure PG6 alternate function to SHRTIMER
11:0	Reserved	Must be kept at reset value.

9. Cyclic redundancy checks management unit (CRC)

9.1. Overview

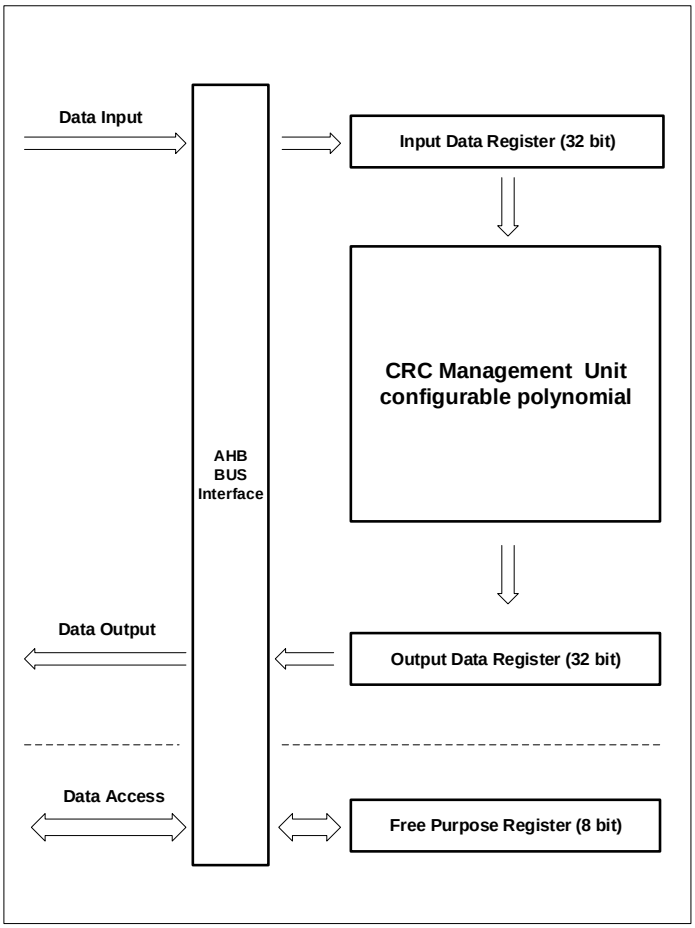
A cyclic redundancy check management (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.

This CRC management unit can be used to calculate 7/8/16/32 bit CRC code within user configurable polynomial.

9.2. Characteristics

- Supports 7/8/16/32 bit data input
- For 7 (8)/16/32 bit input data length, the calculation cycles are 1/2/4 AHB clock cycles
- User configurable polynomial value and size
- After CRC module·reset, user can configure initial value
- Free 8-bit register is unrelated to calculation and can be used for any other goals by any other peripheral devices

Figure 9-1. Block diagram of CRC management unit



9.3. Function overview

- CRC management 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 setting the CRC_CTL register, the new input raw data will be calculated based on the result of previous value of CRC_DATA.

  CRC management will spend 4/2/1 AHB clock cycles for 32/16/8(7) bit data size. During this period, AHB will not be hanged because of the existence of the 32bit input buffer.

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

  CRC_FDATA is unrelated to the CRC calculation. Independent read and write operations can be performed at any time.

- Reversible function can reverse the input data and output data.

  For input data, 3 reverse types can be selected.

  Original data is 0x3456CDEF:

  1) byte reverse:

  32-bit data is divided into 4 groups and reverse implement in group inside. Reversed data: 0x2C6AB3F7

  2) half-word reverse:

  32-bit data is divided into 2 groups and reverse implement in group inside. Reversed data: 0x6A2CF7B3

  3) word reverse:

  32-bit data is divided into 1 groups and reverse implement in group inside. Reversed data: 0xF7B36A2C

  For output data, reverse type is word reverse.

  For example: when REV_O=1, calculation result 0x3344CCDD will be converted to 0xBB3322CC.

- User configurable initial calculation data is available.

  When RST bit is set or write operation to CRC_IDATA register, the CRC_DATA register will be automatically initialized to the value in CRC_IDATA.

- User configurable polynomial.
- Depends on PS[1:0] bits, the valid polynomial and output bit width can be selected by user. If the polynomial is less than 32 bit, the high bits of the input data and output data is unavailable. It is strongly recommend resetting the CRC management unit after change the PS[1:0] bits or polynomial.

9.4. Register definition

CRC base address: 0x4002 3000

9.4.1. 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. Write value cannot be read because the read value is the previous CRC calculation result.

9.4.2. 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	Must be kept 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

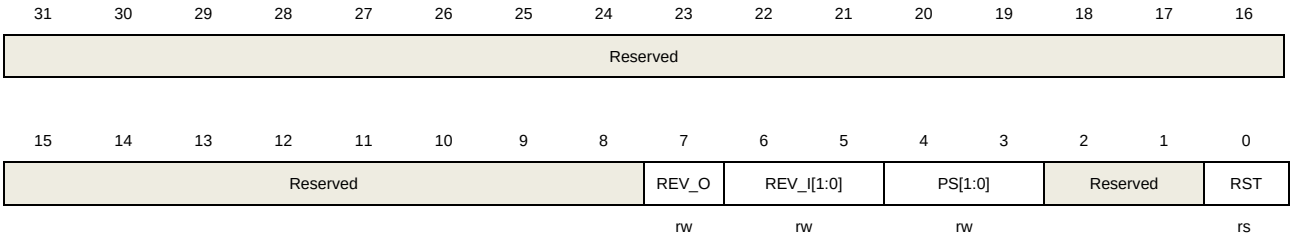
by any other peripheral. The CRC_CTL register will generate no effect to the byte.

9.4.3. Control register (CRC_CTL)

Address offset: 0x08

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	REV_O	Reverse output data value in bit order 0:Not bit reversed for output data 1:Bit reversed for output data
6:5	REV_I[1:0]	Reverse type for input data 0: Dot not use reverse for input data 1: Reverse input data with every 8-bit length 2: Reverse input data with every 16-bit length 3: Reverse input data with whole 32-bit length
4:3	PS[1:0]	Size of polynomial 0: 32 bit 1: 16 bit (POLY [15:0] is used for calculation.) 2: 8 bit (POLY [7:0] is used for calculation.) 3: 7 bit (POLY [6:0] is used for calculation.)
2:1	Reserved	Must be kept at reset value.
0	RST	Software writes and reads. Set this bit can reset the CRC_DATA register. When set, the value of the CRC_DATA register is automatically initialized to the value in the CRC_IDATA register and then automatically cleared by hardware. This bit will take no effect to CRC_FDATA.

9.4.4. Initialization data register (CRC_IDATA)

Address offset: 0x10

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
IDATA[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	IDATA[31:0]	Configurable initial CRC data value When RST bit in CRC_CTL asserted, CRC_DATA will be programmed to this value.

9.4.5. Polynomial register (CRC_POLY)

Address offset: 0x14

Reset value: 0x04C1 1DB7

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
POLY[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	POLY[31:0]	User configurable polynomial value This value is used together with PS[1:0] bits.

10. Trigonometric Math Unit (TMU)

10.1. Overview

The Trigonometric Math Unit (TMU) is a fully configurable block that execute common trigonometric and arithmetic operations.

The TMU calculation unit can be used to calculate total 9 kinds of operations. The operation data must meet IEEE 32-Bit Single Precision Floating-Point Format.

10.2. Characteristics

- Input data and result support 32-Bit Single Precision Floating-Point Format
- 9 different operation modes

Table 10-1. 9 different operation modes

Mode Number	Operation
0	R0 =x * 2π
1	R0 = x/2π
2	R0 = $\sqrt{x}$
3	R0 =sin (x)
4	R0 = cos (x)
5	R0 = arctan (x)
6	R0 = Ratio of X & Y, R1 = Quadrant value (0.0, ±0.25, ±0.5)
7	R0 = x/y
8	R0 = $\sqrt{x^2+y^2}$

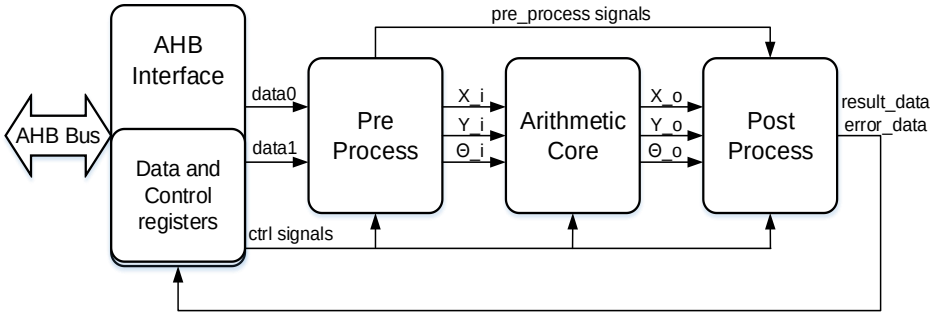
- For mode 0 and 1, operation finishes in 4 clock cycles. For others mode, 7 clock cycles is needed
- The overflow and underflow error flag can be read in the status register
- Optional operation finish interrupt

10.3. Function overview

10.3.1. TMU block diagram

The TMU block diagram show as ***Figure 10-1. Block diagram of Trigonometric Math Unit***.

Figure 10-1. Block diagram of Trigonometric Math Unit



10.3.2. Data format

The operation data and calculation result data format is given in ***Table 10-2. IEEE 32-Bit Single Precision Floating-Point Format***. They must meet IEEE 32 bit Single Precision Floating-Point.

Table 10-2. IEEE 32-Bit Single Precision Floating-Point Format

S [31]	E [30:23]	M [22:0]	Value (V)
0	0	0	Zero (V = 0)
1	0	0	Negative Zero (V = -0)
0 +ve 1 -ve	0	non zero	De-normalized (V=$(-1)^{s}*2^{(-126)}*$ (0.M))
0 +ve 1 -ve	1 to 254	0 to 0x7FFFFF	Normal Range(V=$(-1)^{s}*2^{(E-127)}*$ (1.M))
0	254	0x7FFFFF	Positive Max (V = +Max)
1	254	0x7FFFFF	Negative Max (V = -Max)
0	max=255	0	Positive Infinity (V = +Infinity)
1	max=255	0	Negative Infinity (V = -Infinity)
x	max=255	non zero	Not A Number (V = NaN)

The treatment of the various IEEE floating-point numerical formats for this TMU is given below:

Negative Zero: All TMU operations generate a positive (S=0, E=0, M=0) zero, never a negative zero if the result of the operation is zero. All TMU operations treat negative zero operations as zero.

De-Normalized Numbers: A de-normalized operand (E=0, M!=0) input is treated as zero (E=0, M=0) by all TMU operations. TMU operations never generate a de-normalized value.

Underflow: Underflow occurs when an operation generates a value that is too small to represent in the given floating-point format. Under such cases, a zero value is returned. If a TMU operation generates an underflow condition, then the latched underflow flag (UDRF) is set to 1. The UDRF flag will remain latched until the next new operation is started.

Overflow: Overflow occurs when an operation generates a value that is too large to represent in the given floating-point format. Under such cases, a positive or negtive Infinity value is

returned. If a TMU operation generates an overflow condition, then the latched overflow flag (OVRF) is set to 1. The OVRF flag will remain latched until the next new operation is started.

Rounding: There are various rounding formats supported by the IEEE standard. Rounding has no meaning for TMU operations (rounding is inherent in the implementation). Hence rounding mode is ignored by TMU operations.

Infinity and Not a Number (NaN): An NaN operand (E=max, M!=0) input is treated as Infinity (E=max, M=0) for all operations. TMU operations will never generate a NaN value but Infinity instead.

10.3.3. Mode 0 description

This operation is equivalent as R0 = x * 2π. x is the input operation data, R0 is the calculation result.

This operation is used in converting per-unit values to radians. Per-unit values are used in control applications to represent normalized radians. If mode 0 is used in converting per-unit values to radians, input data should meet the range of [-1, 1]:

Table 10-3. Convert per-unit values to radians in Mode 0

Per-unit	Radians
1.0	2π
0.0	0
-1.0	-2π

This mode only has OVRF flag and UDRF =0. The OVRF condition is as below:

If R0 result is too big for floating-point number (E > 255), R0 = ±Infinity, OVRF = 1.

10.3.4. Mode 1 description

This operation is equivalent as R0 = x / 2π. x is the input operation data, R0 is the calculation result.

This operation is used in converting radians to per-unit values. Per unit values are used in control representing normalized Radians. If mode 1 is used in converting radians to per-unit values, input data should meet the range of [-2π, 2π]:

Table 10-4. Convert radians values to per-unit values in Mode 1

Per-unit	Radians
1.0	2π
0.0	0
-1.0	-2π

This mode only has UDRF flag and OVRF =0. The UDRF condition is as below:

If R0 result is too small for floating-point number (E < 0), R0 = 0.0, UDRF = 1.

10.3.5. Mode 2 description

This operation is equivalent as R0 =$\sqrt{x}$. x is the input operation data, R0 is the calculation result.

This mode only has OVRF flag and UDRF =0. The OVRF condition is as below:

```
/* Check if input is negative */
If( x < 0.0 or x == -Infinity ) {
    /* Return zero */
    S = 0;
    E = 0;
    M = 0;
    /* Set overflow flag */
    OVRF = 1;
}
If( x == +Infinity ) {
    /* Return Infinity */
    S = 0;
    E = 255;
    M = 0;
    /* Set overflow flag */
    OVRF = 1;
}
```

10.3.6. Mode 3 description

This mode performs the following equivalent operation:

1. Make PerUnit equal to the fraction of x, PerUnit = fraction(x).
2. R0 = sin(PerUnit * 2π).

In control applications radians are usually normalized to the range of -1.0 to 1.0. It means that the range of PerUnit * 2π values is (-2π, 2π).

So only the fraction part of operation data x is used in this mode. The x whole part has no effect on the result. This equates to the sine waveform repeating itself every 2π radians.

This mode has no neither UDRF nor OVRF. If the result is too small, the result will return 0.

10.3.7. Mode 4 description

This mode performs the following equivalent operation:

1. Make PerUnit equal to the fraction of x, PerUnit = fraction(x).
2. R0 = cos(PerUnit * 2π).

In control applications radians are usually normalized to the range of -1.0 to 1.0. It means that the range of PerUnit * 2π values is (-2π, 2π).

So only the fraction part of operation data x is used in this mode. The x whole part has no effect on the result. This equates to the cosine waveform repeating itself every 2π radians.

This mode has no neither UDRF nor OVRF. If the result is too small, the result will return 0.

10.3.8. Mode 5 description

This mode computes the arc tangent of a given value and returns the result as a per-unit value: R0 = PerUnit = arctan(x)/2π.

The operation limits the input range of the input value x to [-1, 1].

Values outside this range return 0.125 as follows:

Table 10-5. The range of input value and R0 value in Mode 5

x	Per Unit	Radians	R0 Value	OVRF Flag
>1.0	0.125	pi/4	0.125	1
1.0	0.125	pi/4	0.125	0
0	0	0	0	0
-1.0	-0.125	-pi/4	-0.125	0
<-1.0	-0.125	-pi/4	-0.125	1

This mode only has OVRF flag and UDRF =0. The OVRF condition is that x outside its range [-1,1].

10.3.9. Mode 6 description

This operation, in conjunction with arctan(x), is used in calculating arctan(x) for a full circle, **full circle arctan(x)** = R1 + arctan(R0) :

X = x value.

Y = y value.

R0 = Ratio of X & Y.

R1 = Quadrant value (0.0, ±0.25, ±0.5).

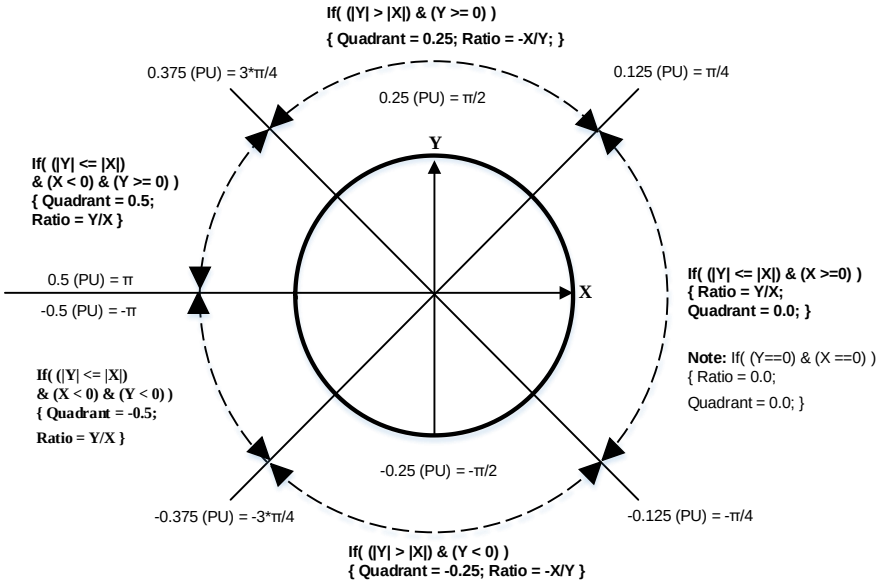
The algorithm for this mode is as follows:

```
if ( ( fabs(Y) == 0.0 ) & ( fabs(X) == 0.0 ) ) {
      R1( Quadrant ) = 0.0;
      R0( Ratio ) = 0.0;
}else if ( fabs(Y) < = fabs(X) ) {
      R0( Ratio ) = Y / X;
      if( X >= 0.0 )
            R1( Quadrant ) = 0.0;
```

```
        else {
        if( Y >= 0.0 )
            R1( Quadrant ) = 0.5;
        else
            R1( Quadrant ) = -0.5;
        }
}else {
        R0( Ratio ) = - X / Y;
        if( Y >= 0.0 )
            R1( Quadrant ) = 0.25;
        else
            R1( Quadrant ) = -0.25;
}
```

The *Figure 10-2. Calculation of R1 (Quadrant) and R0 (Ratio) Based on Y and X Values* shows how the values R0 and R1 are generated based on the contents of X and Y.

Figure 10-2. Calculation of R1 (Quadrant) and R0 (Ratio) Based on Y and X Values



This mode has both UDRF and OVRF. The UDRF and OVRF condition is as below table:

Table 10-6. The condition of UDRF and OVRF flag in Mode 6

Division(Ratio of X & Y)	R0	OVRF	UDRF
0/0	0	1	0
0/Inf	0	0	1
Inf/Inf	inf	0	1
Normal/Inf	0	0	1

10.3.10. Mode 7 description

This operation is equivalent as R0 = x/y. x and y is the input operation data, R0 is the calculation result.

This mode has both UDRF and OVRF. The UDRF and OVRF condition is as below table:

Table 10-7. The condition of UDRF and OVRF flag in Mode 7

Division(X/Y)	R0	OVRF	UDRF
0/0	0	1	0
0/Inf	0	0	1
Inf/Normal	inf	1	0
Inf/0	inf	1	0
Inf/Inf	inf	0	1
Normal/0	inf	1	0
Normal/Inf	0	0	1

10.3.11. Mode 8 description

This operation is equivalent as R0 = $\sqrt{x^2+y^2}$. x and y is the input operation data, R0 is the calculation result.

This mode only has OVRF flag and UDRF =0. The OVRF condition is as below:

If R0 result is too big for floating-point number (E > 255), R0 = +Infinity, OVRF = 1.

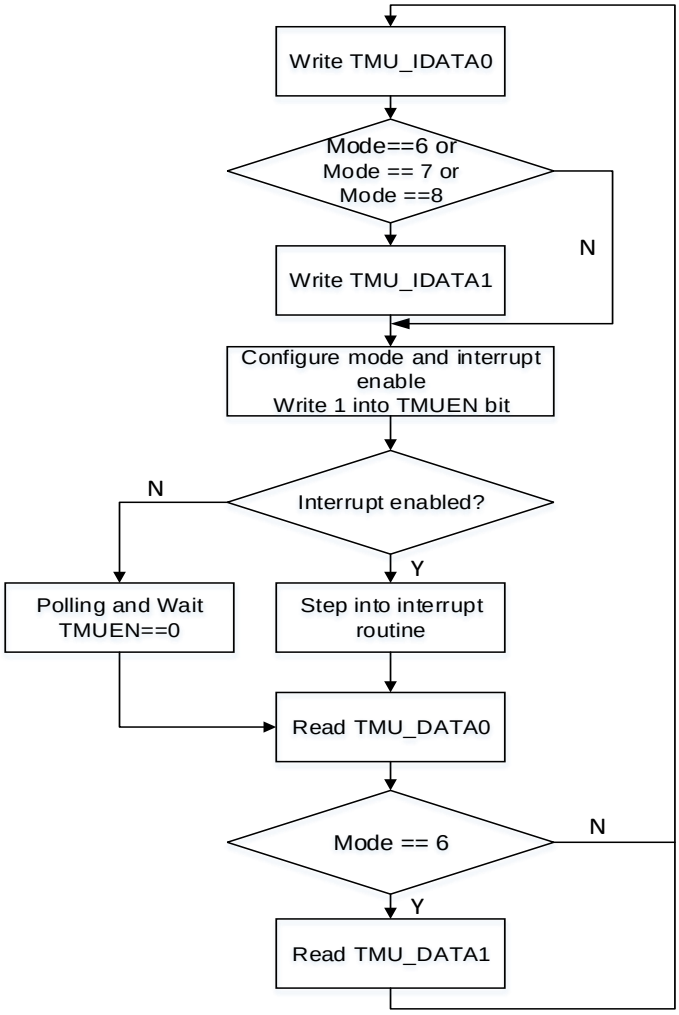
10.4. Software guideline

It is recommended to use the following procedure to read back the TMU result.

1. Write TMU_IDATA0 register, if mode is 6 or 7 or 8 the TMU_IDATA1 register should also be written.
2. Configure the operation mode and interrupt enable bit, write the TMUEN bit to start the calculation.
3. The interrupt will be asserted as soon as the calculation finishes if the interrupt enable is 1, otherwise the software should poll the TMUEN bit and wait until the TMUEN bit is 0.
4. Read the TMU_DATA0 register, if the mode is 6 the TMU_DATA1 register should also be read.
5. Read the UDRF and OVRF bit to guarantee no error occurs.

TMU program guidline show as ***Figure 10-3. TMU program guidline***.

Figure 10-3. TMU program guidline



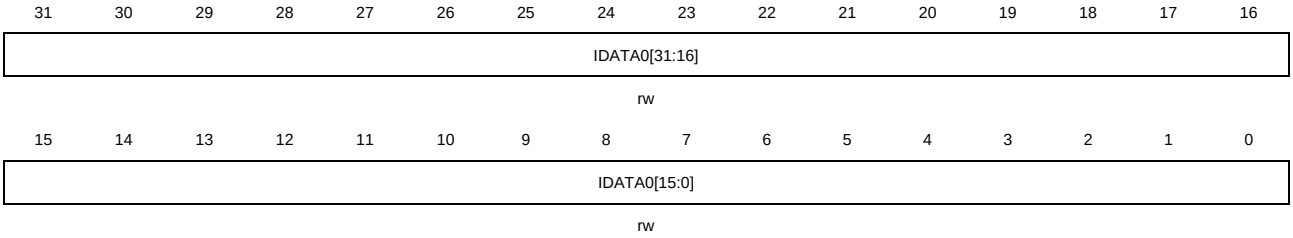
10.5. TMU register

TMU base address: 0x4008 0000

10.5.1. Input data0 register (TMU_IDATA0)

Address offset: 0x00
Reset value: 0x3F80 0000

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

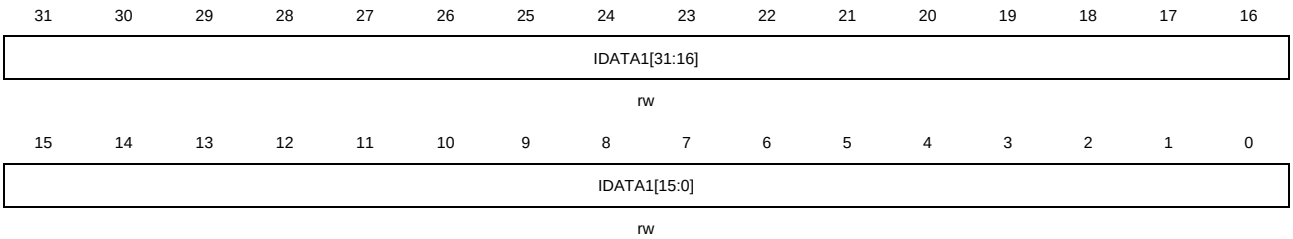


Bits	Fields	Descriptions
31:0	IDATA0[31:0]	The value of input data Mode0~5: IDATA0 is the only operation data Mode6: IDATA0 is the X value Mode7: IDATA0 is the dividend Mode8: IDATA0 is the X value or Y value IDATA0 must meet IEEE 32-Bit Single Precision Floating-Point Format.

10.5.2. Input data1 register (TMU_IDATA1)

Address offset: 0x04
Reset value: 0x3F80 0000

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



Bits	Fields	Descriptions
31:0	IDATA1[31:0]	The value of input data Mode0~5: IDATA1 is ignored mode6: IDATA1 is the Y value

mode7: IDATA1 is the divisor

mode8: IDATA1 is the X value or Y value

IDATA1 must meet IEEE 32-Bit Single Precision Floating-Point Format.

10.5.3. Control register (TMU_CTL)

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									CFIF	CFIE		MODE			TMUEN
									ro	rw		rw			rw

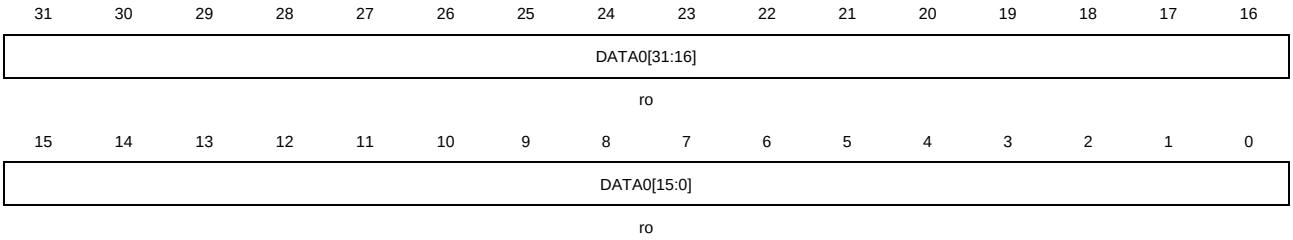
Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6	CFIF	Calculation finish interrupt flag When CFIE is enable, this bit and interrupt signal will be asserted 1 as soon as the calculation finished. And this bit and interrupt signal will be asserted 0 if TMU_DATA0 or TMU_DATA1 or TMU_STAT is read.
5	CFIE	The bit of enable calculation finish interrupt 1: enable calculation finish interrupt 0: disable calculation finish interrupt
4:1	MODE[3:0]	Set the mode of TMU: 0000: R0 = x * 2π 0001: R0 = x/2π 0010: R0 = $\sqrt{x}$ 0011: R0 = sin(x) 0100: R0 = cos(x) 0101: R0 = arctan(x) 0110: R0 = Ratio of X & Y, R1 = Quadrant value 0111: R0 = x/y 1000: R0 = $\sqrt{x^2+y^2}$ 1001~1111: reserved
0	TMUEN	The bit of enable TMU Write 1 to start TMU module calculation. Hardware clear this bit to 0 if calculation finished.

10.5.4. Data0 register (TMU_DATA0)

Address offset: 0x0C

Reset value: 0x3400 0000

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



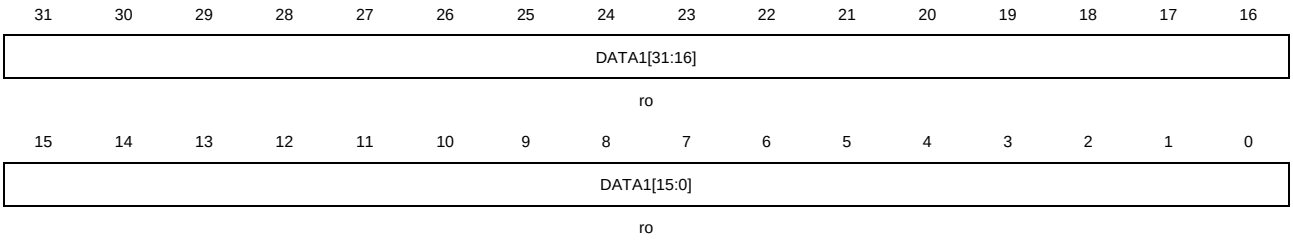
Bits	Fields	Descriptions
31:0	DATA0[31:0]	The result of calculation Mode 0~5,7,8: TMU_DATA0 is the only result value Mode6: TMU_DATA0=Ratio of X and Y TMU_DATA0 must meet IEEE 32-Bit Single Precision Floating-Point Format.

10.5.5. Data1 register (TMU_DATA1)

Address offset: 0x10

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:0	DATA1[31:0]	The result of calculation Mode 0~5,7,8: TMU_DATA1 is ignored Mode6: TMU_DATA1= Quadrant value (0.0, ±0.25, ±0.5) TMU_DATA1 must meet IEEE 32-Bit Single Precision Floating-Point Format.

10.5.6. Status register (TMU_STAT)

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
Reserve														UDRF	OVRF
														ro	ro

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	UDRF	The flag of underflow 0: No underflow 1: Underflow This bit is set and cleared by hardware. when the next TMU calculation is started, this bit is cleared by hardware.
0	OVRF	The flag of overflow 0: No overflow 1: Overflow This bit is set and cleared by hardware. when the next TMU calculation is started, this bit is cleared by hardware.

11. Direct memory access controller (DMA)

11.1. 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.

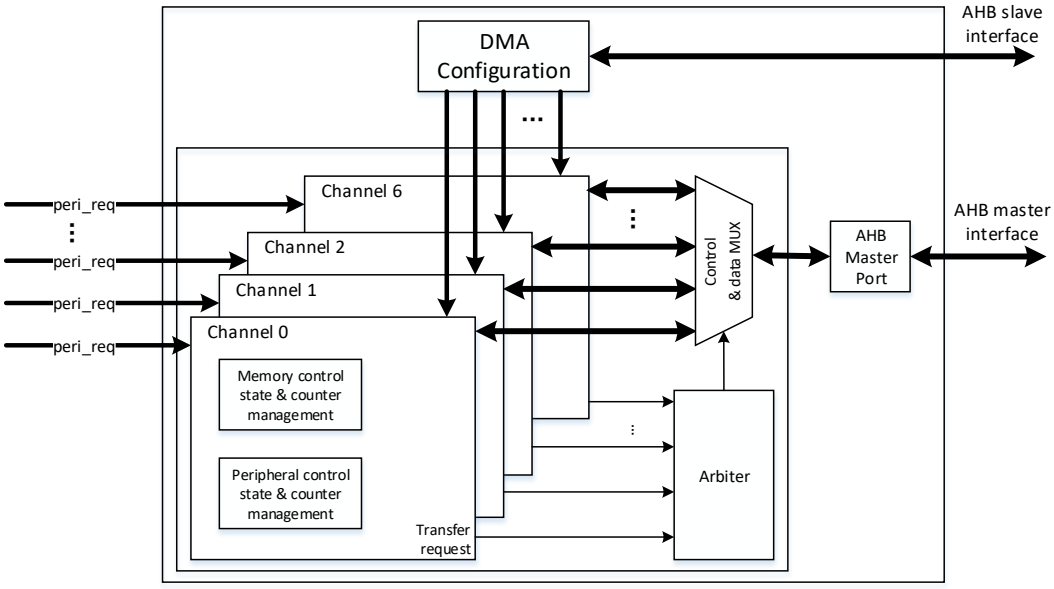
The system bus is shared by the DMA controller and the Cortex®-M33 core. When the DMA and the CPU are targeting the same destination, the DMA access may stop the CPU access to the system bus for some bus cycles. Round-robin scheduling is implemented in the bus matrix to ensure at least half of the system bus bandwidth for the CPU.

11.2. 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.

11.3. Block diagram

Figure 11-1. Block diagram of DMA



As shown in ***Figure 11-1. 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

11.4. Function overview

11.4.1. 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 11-1. DMA transfer operation***.

Table 11-1. 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[15:0] @0x0 2: Write B5B4[15:0] @0x2 3: Write B9B8[15:0] @0x4 4: Write BDBC[15:0] @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 CHEvN 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.

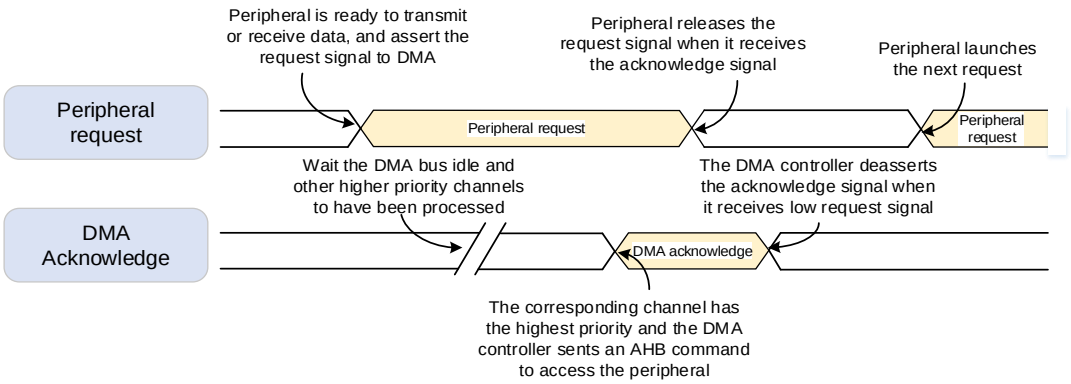
11.4.2. Peripheral handshake

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 11-2. Handshake mechanism shows how the handshake mechanism works between the DMA controller and peripherals.

Figure 11-2. Handshake mechanism



11.4.3. 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.

11.4.4. 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.

11.4.5. 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.

11.4.6. 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.

11.4.7. 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.

11.4.8. 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.

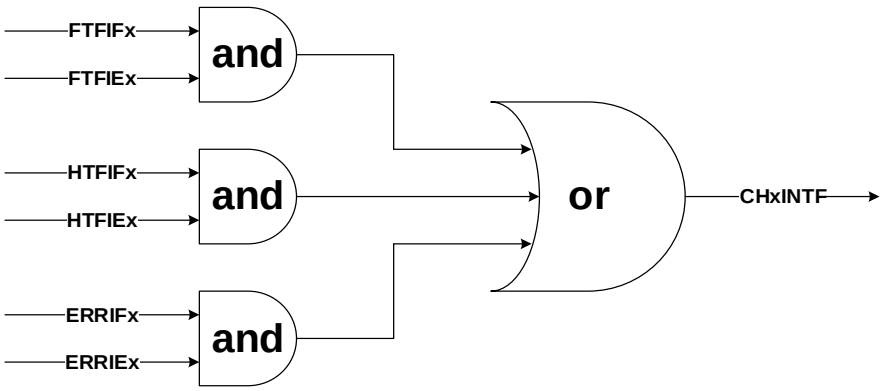
Each interrupt event has a dedicated flag bit in the DMA_INTF register, a dedicated clear bit in the DMA_INTC register, and a dedicated enable bit in the DMA_CHxCTL register. The relationship is described in the following ***Table 11-2. interrupt events***.

Table 11-2. 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 11-3. DMA interrupt logic***, an interrupt can be produced when any type of interrupt event occurs and enabled on the channel.

Figure 11-3. DMA interrupt logic



NOTE: "x" indicates channel number (for DMA0, x=0…6. for DMA1, x=0…4).

11.4.9. 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 11-4. DMA0 request mapping*** and ***Figure 11-5. 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 11-3. DMA0 requests for each channel lists the support request from peripheral for each channel of DMA0, and *Table 11-4. DMA1 requests for each channel* lists the support request from peripheral for each channel of DMA1.

Figure 11-4. DMA0 request mapping

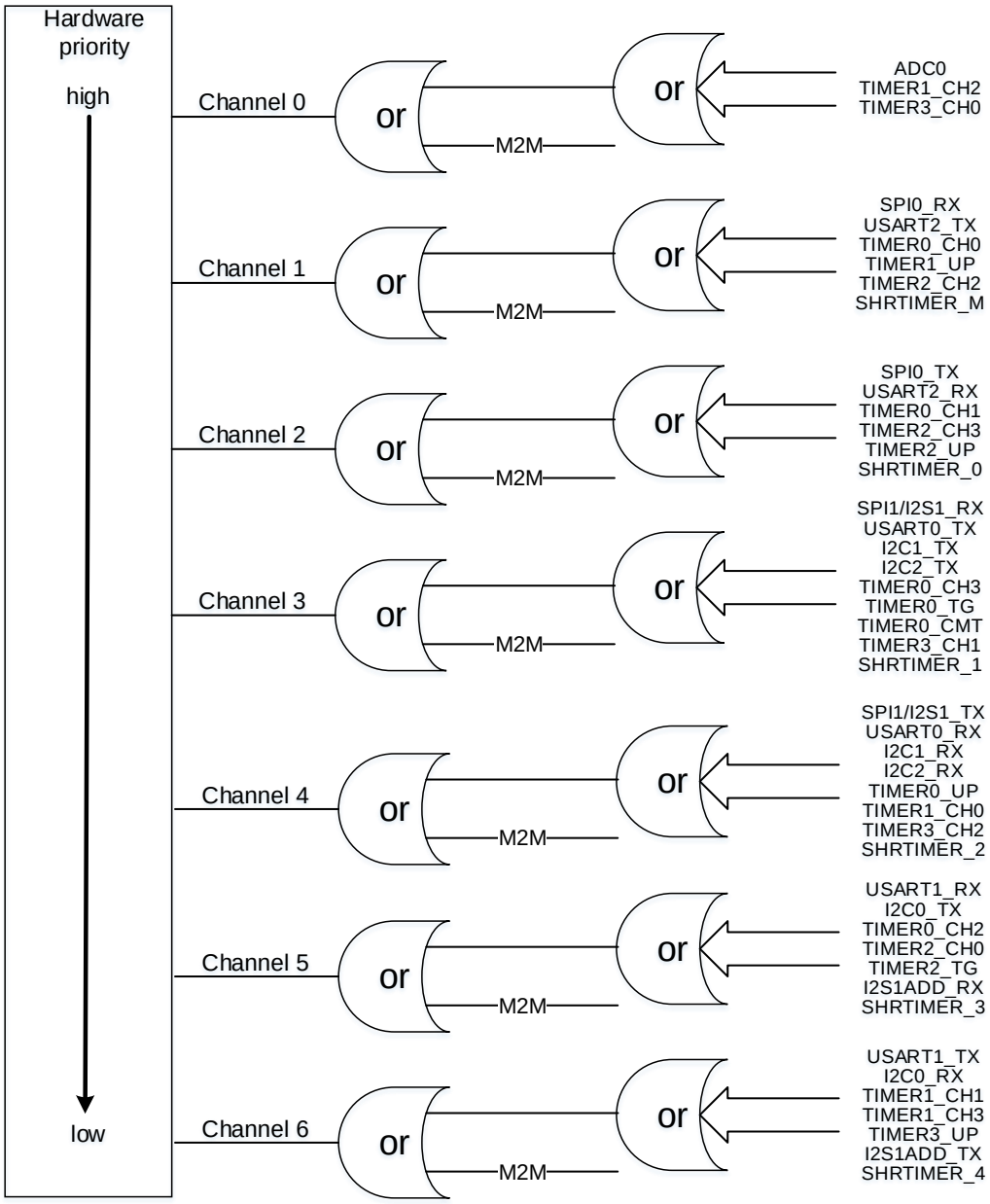


Table 11-3. DMA0 requests for each channel

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

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
TIMER2	•	TIMER2_CH2	TIMER2_CH3 TIMER2_UP	•	•	TIMER2_CH0 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	I2S1ADD_RX	I2S1ADD_TX
USART	•	USART2_TX	USART2_RX	USART0_TX	USART0_RX	USART1_RX	USART1_TX
I2C	•	•	•	I2C1_TX I2C2_TX	I2C1_RX I2C2_RX	I2C0_TX	I2C0_RX
SHRTIMER	•	SHRTIMER_M	SHRTIMER_0	SHRTIMER_1	SHRTIMER_2	SHRTIMER_3	SHRTIMER_4

Figure 11-5. DMA1 request mapping

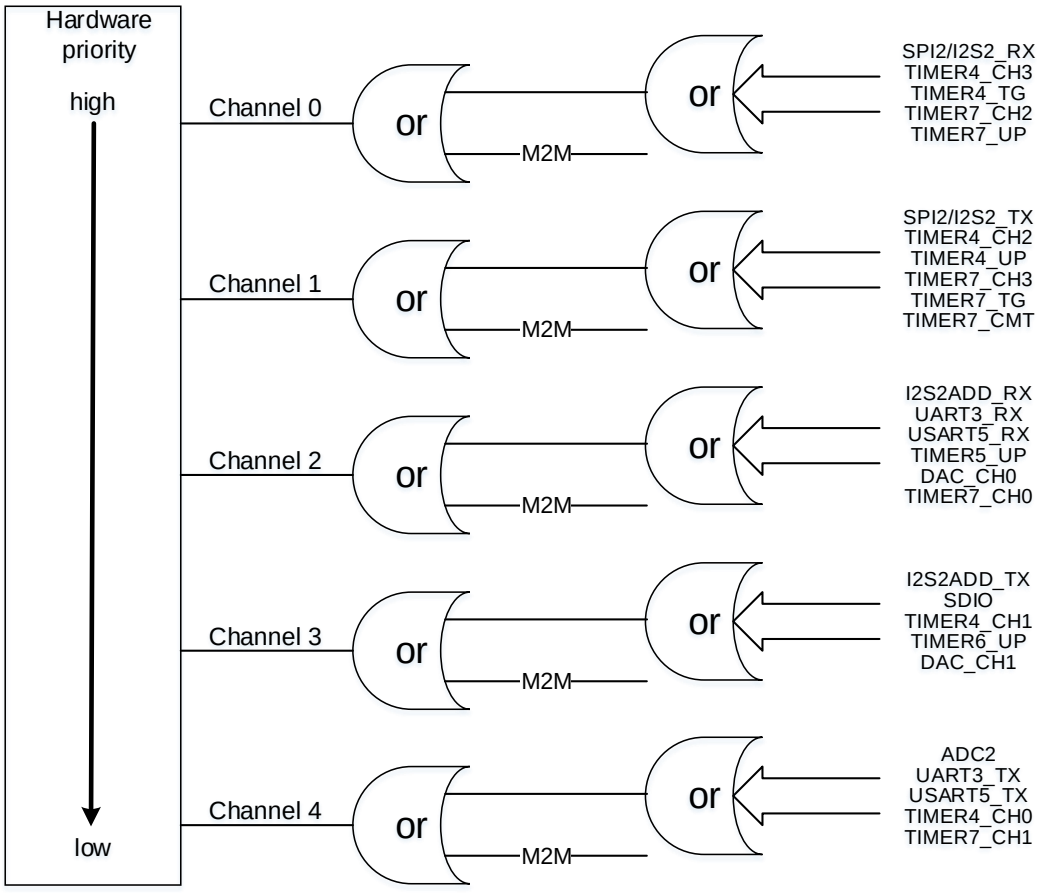


Table 11-4. DMA1 requests for each channel

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_CH3	TIMER7_CH0	•	TIMER7_CH1

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4
	TIMER7_UP	TIMER7_TG TIMER7_CMT			
ADC2	•	•	•	•	ADC2
DAC	•	•	DAC_CH0	DAC_CH1	•
SPI/I2S	SPI2/I2S2_R	SPI2/I2S2_TX	I2S2ADD_RX	I2S2ADD_TX	•
USART	•	•	UART3_RX USART5_RX	•	UART3_TX USART5_TX
SDIO	•	•	•	SDIO	•

11.5. Register definition

DMA0 base address: 0x4002 0000

DMA1 base address: 0x4002 0400

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

11.5.1. Interrupt flag register (DMA_INTF)

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				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	Must be kept 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.

11.5.2. Interrupt flag clear register (DMA_INTC)

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				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	Must be kept 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.

11.5.3. Channel x control register (DMA_CHxCTL)

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

Address offset: 0x08 + 0x14 × x

Reset value: 0x0000 0000

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	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	Must be kept 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’.

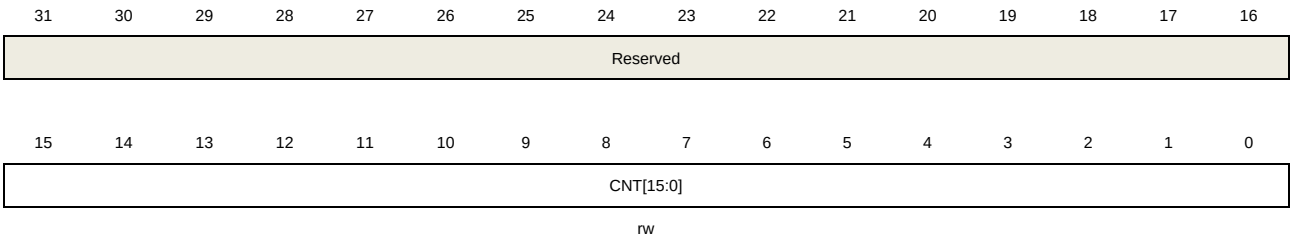
5	CMEN	Circular mode enable Software set and cleared 0: Disable circular mode 1: Enable circular mode This bit can not be written when CHEN is ‘1’.
4	DIR	Transfer direction Software set and cleared 0: Read from peripheral and write to memory 1: Read from memory and write to peripheral This bit can not 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

11.5.4. Channel x counter register (DMA_CHxCNT)

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

Address offset: 0x0C + 0x14 × x
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:16	Reserved	Must be kept 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.

11.5.5. 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

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

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.

11.5.6. 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

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

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

12. Debug (DBG)

12.1. Introduction

The GD32E50x 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®-M33. 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®-M33 Technical Reference Manual
- Arm Debug Interface v5 Architecture Specification

The DBG hold unit helps debugger to debug power saving mode, TIMER, I2C, WWDGT, and 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.

12.2. JTAG/SW function description

Debug capabilities can be accessed by a debug tool via serial wire (SW - Debug Port) or JTAG interface (JTAG - Debug Port).

12.2.1. 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.

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

12.2.3. JTAG daisy chained structure

The Cortex®-M33 JTAG TAP is connected to a boundary-scan (BSD) JTAG TAP. The BSD JTAG IR is 5-bit width, while the Cortec-M33 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 Cortex®-M33 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.

12.2.4. Debug reset

The JTAG-DP and SW-DP registers are in the power on reset domain. The system reset initializes the majority of the Cortex®-M33, 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.

12.2.5. JEDEC-106 ID code

The Cortex®-M33 integrates JEDEC-106 ID code, which is located in ROM table and mapped on the address of 0xE00FF000_0xE00FFFFF.

12.3. Debug hold function description

12.3.1. 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.

12.3.2. 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.

12.4. DBG registers

DEBUG base address: 0xE0044000

12.4.1. ID code register (DBG_ID)

Address offset: 0x00

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.

12.4.2. 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
SHRTIMER_HOLD.	TIMER10_HOLD	TIMER9_HOLD	TIMER8_HOLD	TIMER13_HOLD	TIMER12_HOLD	TIMER11_HOLD	Reserved		I2C2_HOLD	CAN1_HOLD	TIMER6_HOLD	TIMER5_HOLD	TIMER4_HOLD	TIMER7_HOLD	I2C1_HOLD
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
I2C0_HOLD	CAN0_HOLD	TIMER3_HOLD	TIMER2_HOLD	TIMER1_HOLD	TIMER0_HOLD	WWDGT_HOLD	FWDGT_HOLD	Reserved		TRACE_IOEN	Reserved	CAN2_HOLD	STB_HOLD	DSLP_HOLD	SLP_HOLD
rw	rw	rw	rw	rw	rw	rw	rw			rw		rw	rw	rw	rw

Bits	Fields	Descriptions
31	SHRTIMER_HOLD	SHRTIMER hold bit This bit is set and reset by software. 0: no effect 1: Hold the SHRTIMER counter for debug when core halted.
30	TIMER10_HOLD	TIMER10 hold bit This bit is set and reset by software. 0: no effect

		1: Hold the TIMER10 counter for debug when core halted.
29	TIMER9_HOLD	TIMER9 hold bit This bit is set and reset by software. 0: no effect 1: Hold the TIMER9 counter for debug when core halted.
28	TIMER8_HOLD	TIMER8 hold bit This bit is set and reset by software. 0: no effect 1: Hold the TIMER8 counter for debug when core halted.
27	TIMER13_HOLD	TIMER13 hold bit This bit is set and reset by software. 0: no effect 1: Hold the TIMER13 counter for debug when core halted
26	TIMER12_HOLD	TIMER12 hold bit This bit is set and reset by software. 0: no effect 1: Hold the TIMER12 counter for debug when core halted.
25	TIMER11_HOLD	TIMER11 hold bit This bit is set and reset by software. 0: no effect 1: Hold the TIMER11 counter for debug when core halted.
24:23	Reserved	Must be kept at reset value.
22	I2C2_HOLD	I2C2 hold bit This bit is set and reset by software. 0: no effect 1: Hold the I2C2 SMBUS timeout for debug when core halted.
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	TIMER3 hold bit This bit is set and reset by software. 0: no effect 1: Hold the TIMER3 counter for debug when core halted.
12	TIMER2_HOLD	TIMER2 hold bit This bit is set and reset by software. 0: no effect 1: Hold the TIMER2 counter for debug when core halted.
11	TIMER1_HOLD	TIMER1 hold bit This bit is set and reset by software. 0: no effect 1: Hold the TIMER1 counter for debug when core halted.
10	TIMER0_HOLD	TIMER0 hold bit This bit is set and reset by software. 0: no effect 1: Hold the TIMER0 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	Reserved	Must be kept at reset value.
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	Reserved	Must be kept at reset value.
3	CAN2_HOLD	CAN2 hold bit This bit is set and reset by software. 0: no effect 1: The receive register of CAN2 stops receiving data when core halted.
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.

13. Analog-to-digital converter (ADC)

13.1. Introduction

A 12-bit successive approximation analog-to-digital converter module(ADC) is integrated on the MCU chip, which can sample analog signals from 16 external channels and 2 internal channels. The 18 ADC sampling channels all support a variety of operation modes. After sampling and conversion, the conversion results can be stored in the corresponding data registers according to the least significant bit alignment(LSB) or the most significant bit alignment(MSB). An on-chip hardware oversample scheme improves performances while off-loading the related computational burden from the MCU.

13.2. Characteristics

- High performance:
  - ADC sampling rsolution: 12-bit, 10-bit, 8-bit or 6-bit resolution.
  - Foreground calibration function.
  - Programmable sampling time.
  - Data mode: the most significant bit and the least significant bit.
  - 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.
- Operation modes:
  - Converts a single channel or scans a sequence of channels.
  - Single operation mode converts selected inputs once per trigger.
  - Continuous operation mode converts selected inputs continuously.
  - Discontinuous operation mode.
  - SYNC mode(the device with two or more ADCs).
- Conversion result threshold monitor function: analog watchdog.
- Interrupt generation:
  - At the end of routine conversions.
  - Analog watchdog event.
- Oversampler:
  - 16-bit data register.
  - Oversampling ratio adjustable from 2 to 256x.
  - Programmable data shift up to 8-bit.

- Module supply requirements: the typical power supply voltage is 3.3V:
  - 1.62V to 2.4V, with ADC maximum frequency is 14MHz.
  - 2.4V to 3.6V, with ADC maximum frequency is 35MHz.
- Channel input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$.

13.3. Pins and internal signals

Figure 13-1. ADC module block diagram (for ADC0 and ADC1) and *Figure 13-2. ADC module block diagram (for ADC2)* show the ADC block diagram and *Table 13-2. ADC input pins definition* gives the ADC pin description.

Table 13-1. ADC internal input signals

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

Table 13-2. ADC input pins definition

Name	Remarks
V_{DDA}	Analog power supply equal to V_{DD}, 1.62 V ≤ V_{DDA}≤ 2.4 V (with ADC maximum frequency is 14MHz) 2.4 V ≤ V_{DDA}≤ 3.6 V (with ADC maximum frequency is 35MHz)
V_{SSA}	Ground for analog power supply equal to V_{SS}
V_{REF+}	The positive reference voltage for the ADC, 1.62 V ≤ V_{REF+} ≤ V_{DDA} (with ADC maximum frequency is 14MHz) 2.4 V ≤ V_{REF+} ≤ V_{DDA} (with ADC maximum frequency is 35MHz)
V_{REF-}	The negative reference voltage for the ADC, V_{REF-} = V_{SSA}
ADCx_IN[15:0]	Up to 16 external channels

13.4. Functional overview

Figure 13-1. ADC module block diagram (for ADC0 and ADC1)

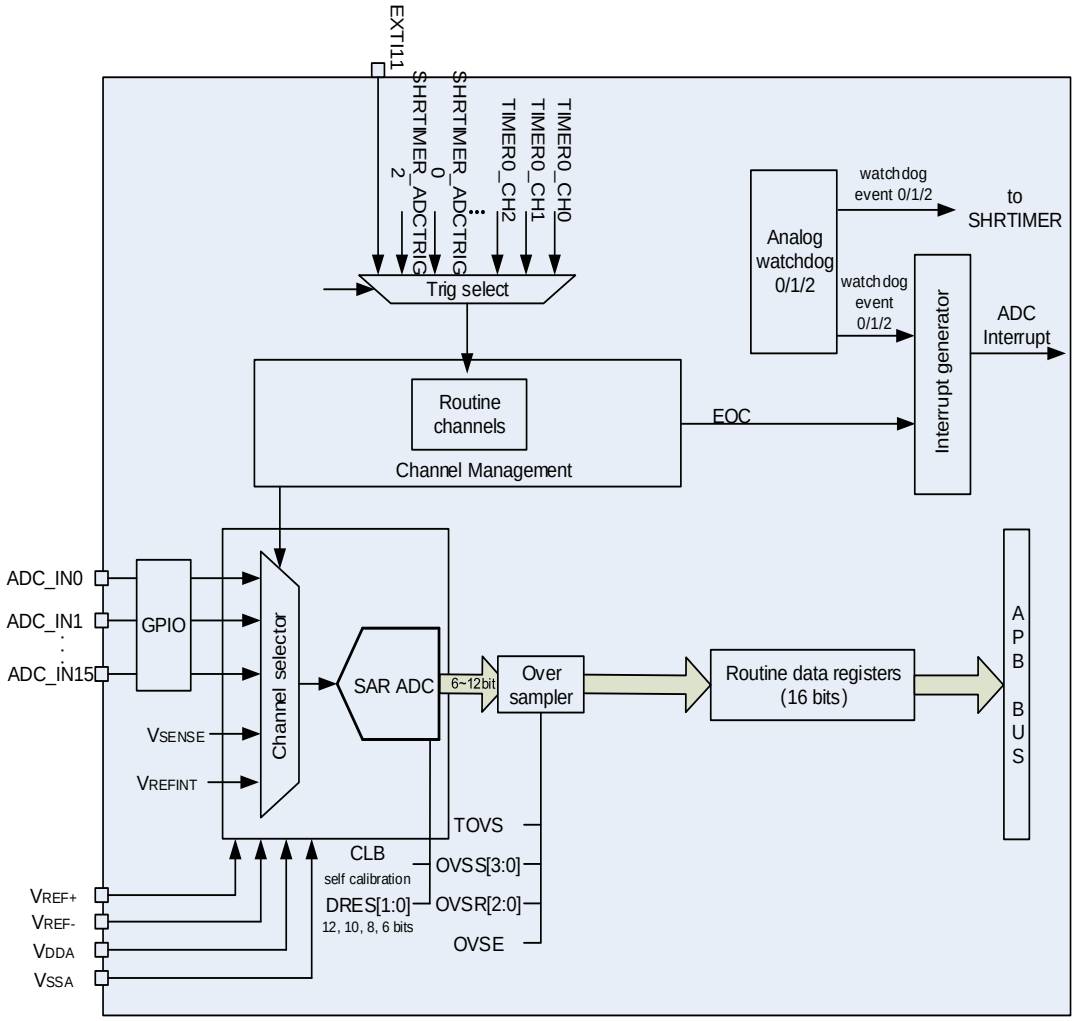
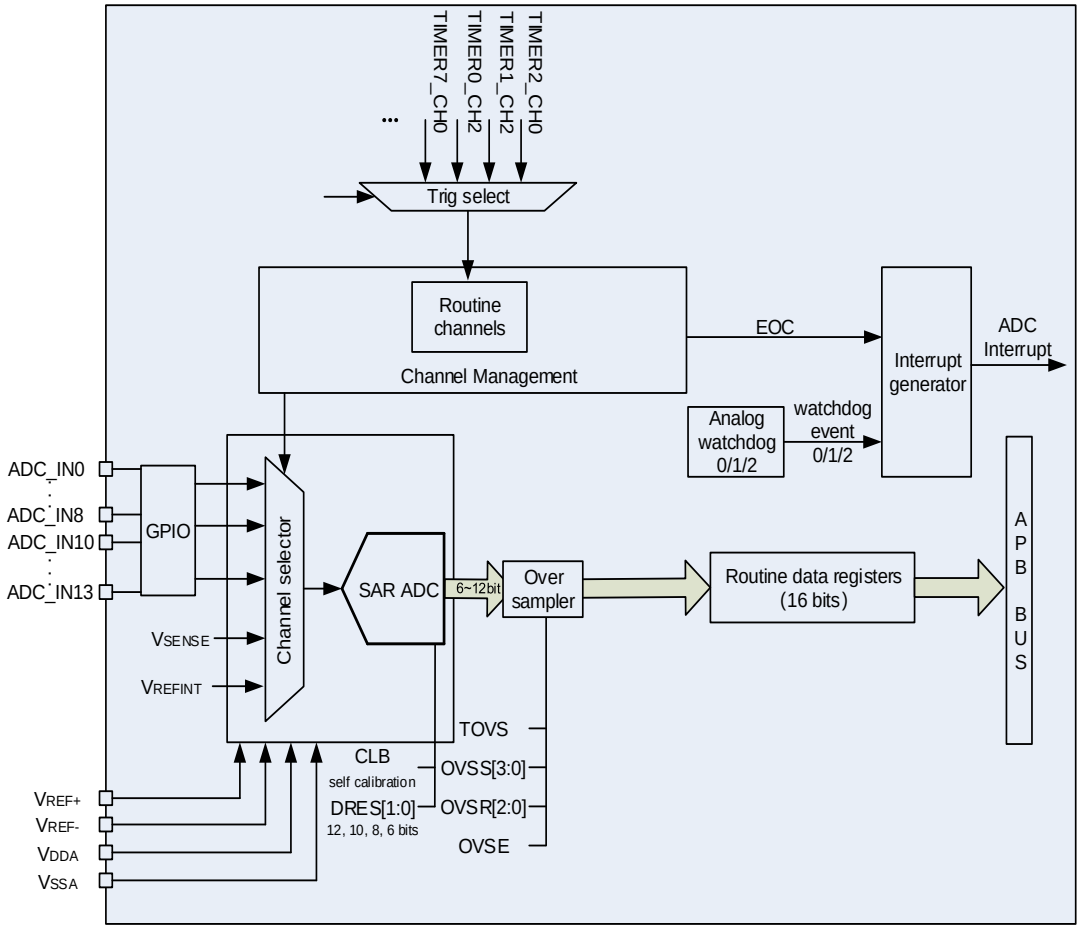


Figure 13-2. ADC module block diagram (for ADC2)



13.4.1. Foreground calibration function

During the foreground calibration 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. The calibration should be performed before starting A/D conversion. The calibration is initiated 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. The calibration process can be repeated by a configurable number of times by setting the CALNUM[2:0] bits in ADC_CTL1 register, and the results will be averaged to get a more precise calibration result.

When the ADC operating conditions change (such as supply power voltage V_{DDA}, positive reference voltage V_{REF+}, 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 CK_ADC to wait for ADC stability.
3. Set CALNUM.

4. Set RSTCLB (optional).
5. Set CLB=1.
6. Wait until CLB=0.

13.4.2. ADC clock

The CK_ADC clock is synchronous with the AHB and APB2 clock and provided by the clock controller. ADC clock can be divided and configured by RCU controller.

13.4.3. ADCON enable

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 put into power-off mode. After ADC is enabled, you need delay t_{SU} time for sampling, the value of t_{SU} please refer to the device datasheet.

13.4.4. Single-ended and differential input channels

By writing into bits DIFCTL[14:0] in the ADC_DIFCTL register, the use can cofigure as differential input or single-ended input,and the ADC must be disabled (ADCON =0) when the user configurate these bits.

The channel n voltage is the difference between positive input and negative input. The positive input is external voltage V_{INn}, and there is a difference of the negative input between single-ended mode and differential input mode. In single-ended input mode, the negative input is V_{REF-}, in differential input mode, the negative input is $V_{IN(n+1)}$. And therefore, channel (n+1) is no longer usable in single-ended mode or in differential mode and must never be configured to be converted.

Channel 15, 16, 17 are forced to single-ended configuration (corresponding bits DIFCTL[n] is always zero), because they are connected to internal channels.

When the channel is used in differential input mode, the input voltages should be differential signals (common mode voltage is V_{REF+}/ 2), and the input ranges are still (V_{REF-} ~ V_{REF+}).

Taking the right-aligned, 12-bit resolution as an example,

1) When V_{INn} is V_{REF+} and $V_{IN(n+1)}$ is V_{REF-}, the conversion result of channel n is 0x0FFF;
2) When V_{INn} is V_{REF-} and $V_{IN(n+1)}$ is V_{REF+}, The conversion result of channel n is 0x0000;
3) When V_{INn} is V_{REF+}/ 2 and $V_{IN(n+1)}$ is V_{REF+}/ 2, the conversion result of channel n is 0x07FF.

D_{out} is the conversion result of channel n, then the differential voltage is：

$$V_{INn}-V_{IN(n+1)} = V_{REF+}*(2*D_{out}/4095-1) \quad (13\text{-}1)$$

13.4.5. Routine sequence

The channel management circuit can organize the sampling conversion channels into a sequence: routine sequence. The routine sequence supports up to 16 channels, and each channel is called routine channel.

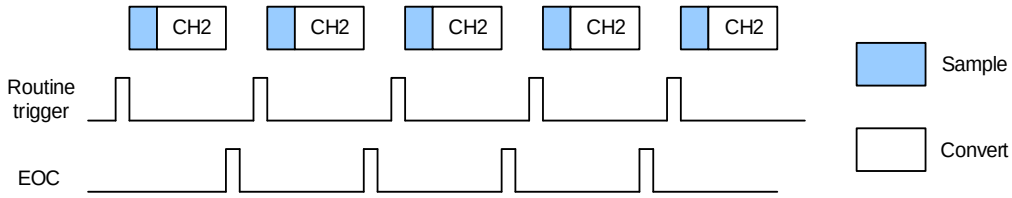
The RL[3:0] bits in the ADC_RSQ0 register specify the total conversion sequence length. The ADC_RSQ0~ADC_RSQ2 registers specify the selected channels of the routine sequence.

13.4.6. Operation modes

Single operation mode

In the single operation mode, the ADC performs conversion on the channel specified in the RSQ0[4:0] bits of ADC_RSQ2 at a routine trigger. 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 13-3. Single operation mode



After conversion of a single routine 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.

Software procedure for a single conversion of a routine 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 routine sequence.
6. Wait the EOC flag to be set.
7. After a delay of one CK_ADC, read the converted date in the ADC_RDATA register.
8. Clear the EOC flag by writing 0 to it.

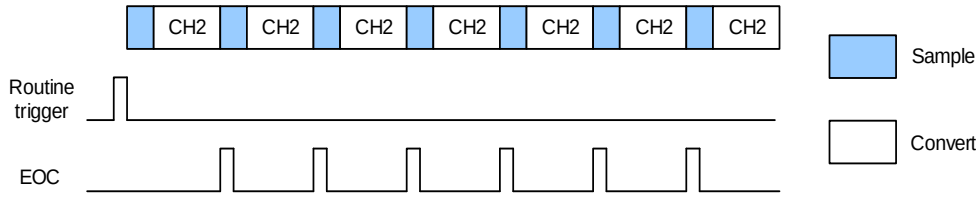
Note: After EOC is set, a delay of one CK_ADC is required before reading the ADC conversion result.

Continuous operation mode

The continuous operation 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 13-4. Continuous operation mode



Software procedure for continuous conversion on a routine 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 routine sequence.
6. Wait the EOC flag to be set.
7. After a delay of one CK_ADC, read the converted date 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.

Note: After EOC is set, a delay of one CK_ADC is required before reading the ADC conversion result.

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

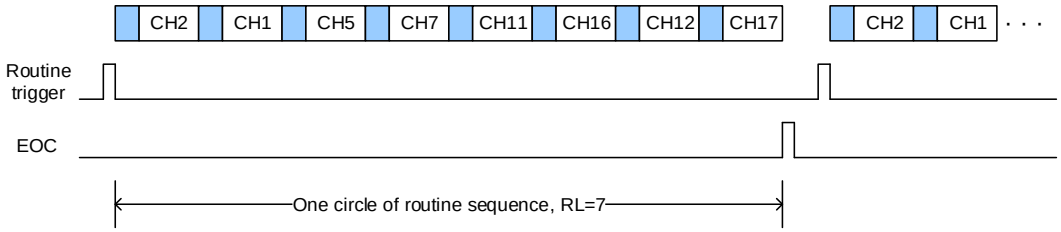
1. Set the CTN and DMA bit in the ADC_CTL1 register.
2. Configure RSQ0 with the analog channel number.
3. Configure ADC_SAMPTx register.
4. Configure ETERC and ETSRC bits in the ADC_CTL1 register if in need.
5. Prepare the DMA module to transfer data from the ADC_RDATA.
6. Set the SWRCST bit, or generate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

Scan operation mode

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

After conversion of a routine sequence, the conversion can be restarted automatically if the CTN bit in the ADC_CTL1 register is set.

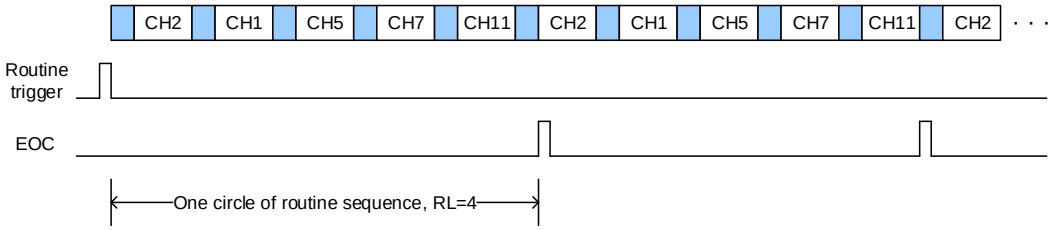
Figure 13-5. Scan operation mode, continuous operation mode disable



Software procedure for scan conversion on a routine sequence:

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 routine sequence.
6. Wait the EOC flag to be set.
7. Clear the EOC flag by writing 0 to it.

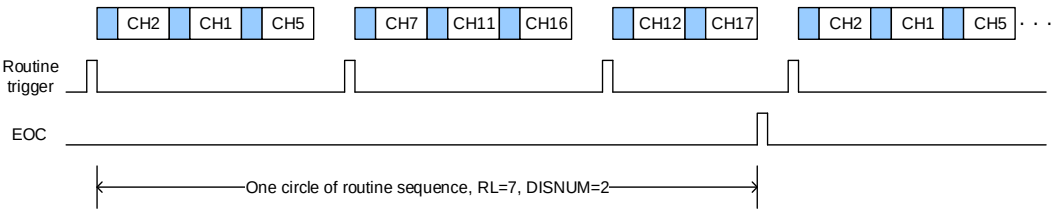
Figure 13-6. Scan operation mode, continuous operation mode enable



Discontinuous operation mode

The discontinuous operation 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 does not exceed 8) which is a part of the sequence selected in the ADC_RSQ0~ADC_RSQ2 registers. The value of n is configured 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 converts the next n channels configured in the ADC_RSQ0~ADC_RSQ2 registers until all the channels of routine sequence channels are done. The EOC will be set after every circle of the routine sequence. An interrupt will be generated if the EOCIE bit is set.

Figure 13-7. Discontinuous operation mode



Software procedure for discontinuous conversion on a routine sequence:

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 routine sequence.
7. Repeat step6 if in need.
8. Wait the EOC flag to be set.
9. Clear the EOC flag by writing 0 to it.

13.4.7. Conversion result threshold monitor

Analog watchdog 0

The analog watchdog 0 is enabled when the RWD0EN bit in the ADC_CTL0 register is set for routine sequence. This function is used to monitor whether the conversion result exceeds the set thresholds, and the WDE0 bit in ADC_STAT register will be set. An interrupt will be generated if the WDE0IE bit is set. The ADC_WDHT0 and ADC_WDLT0 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 RWD0EN, WD0SC and WD0CHSEL [4:0] bits in ADC_CTL0 register, can be monitored by the analog watchdog 0.

Analog watchdog 1/2

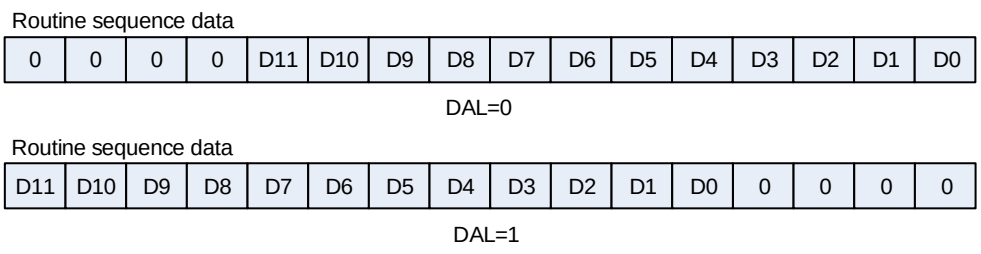
The analog watchdog 1/2 are more flexible, and can configure the watchdog function of single or serveral channels.

The analog watchdog 1 function can be enabled by configuring the corresponding bits in the AWD1CS [17: 0] bits in the ADC_WD1SR register. Similarly, the watchdog 2 function can be configured. The high / low threshold of the analog watchdog 1/2 can be configured in the ADC_WDT1 and ADC_WDT2 registers.

13.4.8. Data storage mode

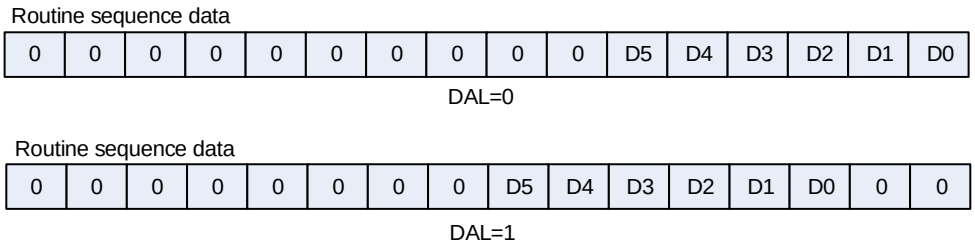
The alignment of data stored after conversion can be specified by DAL bit in the ADC_CTL1 register.

Figure 13-8. 12-bit data storage mode



6-bit resolution data alignment is different from 12-bit/10-bit/8-bit resolution data alignment, shown as ***Figure 13-9. 6-bit data storage mode***

Figure 13-9. 6-bit data storage mode



13.4.9. Sample time configuration

The number of CK_ADC 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” CK_ADC cycles. For example: CK_ADC = 30MHz and sample time is 1.5 cycles, the total conversion time is “1.5+12.5” CK_ADC cycles, that means 0.467us.

13.4.10. External trigger configuration

The conversion of routine sequence can be triggered by rising edge of external trigger inputs. The external trigger source of routine sequence is controlled by the ETSRC[3:0] bits in the ADC_CTL1 register.

Table 13-3. External trigger source for ADC0 and ADC1

ETSRC[3:0]	Trigger Source	Trigger Type
0000	TIMER0_CH0	Hardware trigger
0001	TIMER0_CH1	
0010	TIMER0_CH2	
0011	TIMER1_CH1	
0100	TIMER2_TRGO	
0101	TIMER3_CH3	
0110	EXTI11/ TIMER7_TRGO	
0111	SWRCST	Software trigger
1000	SHRTIMER_ADCTRIG0	Hardware trigger
1001	SHRTIMER_ADCTRIG2	
1010~1111	reserved	-

Table 13-4. External trigger source for ADC2

ETSRC[3:0]	Trigger Source	Trigger Type
0000	TIMER2_CH0	Hardware trigger
0001	TIMER1_CH2	
0010	TIMER0_CH2	

ETSRC[3:0]	Trigger Source	Trigger Type
0011	TIMER7_CH0	
0100	TIMER7_TRGO	
0101	TIMER4_CH0	
0110	TIMER4_CH2	
0111	SWRCST	Software trigger
1000~1111	reserved	-

13.4.11. DMA request

The DMA request, which is enabled by the DMA bit of ADC_CTL1 register, is used to transfer data of routine sequence for conversion of more than one channel. The ADC generates a DMA request at the end of conversion of a routine 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.

13.4.12. ADC internal channels

When the TSVREN bit of ADC_CTL1 register is set, the temperature sensor channel (ADC0_CH16) and V_{REFINT} channel (ADC0_CH17) are 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 ts_temp μs (please refer to the datasheet). 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 the chip production process variation, the internal temperature sensor is more appropriate to 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.

To use the temperature sensor:

1. Configure the conversion sequence (ADC_IN16) and the sampling time(ts_temp μs) for the channel.
2. Enable the temperature sensor by setting the TSVREN bit in the ADC control register 1 (ADC_CTL1).
3. Start the ADC conversion by setting the ADCON bit or by external trigger.
4. Read the temperature data($V_{temperature}$) in the ADC data register, and get the temperature with the following equation:

$$\text{Temperature (°C)} = \{(V_{25} - V_{temperature}) / \text{Avg\_Slope}\} + 25.$$

V_{25}: internal temperature sensor output voltage at 25°C, the typical value please refer to

the datasheet.

Avg_Slope: average slope for curve between temperature vs. internal temperature sensor output voltage, the typical value please refer to the datasheet.

13.4.13. Programmable resolution (DRES)

The resolution is configured by programming the DRES[1:0] bits in the ADC_OVSAMPCTL register. For applications that do not require high data accuracy, lower resolution allows faster conversion time. The DRES[1:0] bits must only be changed when the ADCON bit is reset. Lower resolution reduces the conversion time needed for the successive approximation steps as shown in ***Table 13-5. tCONV timings depending on resolution.***

Table 13-5. t_{CONV} timings depending on resolution

DRES[1:0] bits	t_{CONV} (ADC clock cycles)	t_{CONV}(ns) at f_{ADC}=30MHz	t_{SMPL}(min) (ADC clock cycles)	t_{ADC} (ADC clock cycles)	t_{ADC}(ns) at f_{ADC}=30MHz
12	12.5	417 ns	1.5	14	467 ns
10	10.5	350 ns	1.5	12	400 ns
8	8.5	283 ns	1.5	10	333 ns
6	6.5	217 ns	1.5	8	267 ns

13.4.14. On-chip hardware oversampling

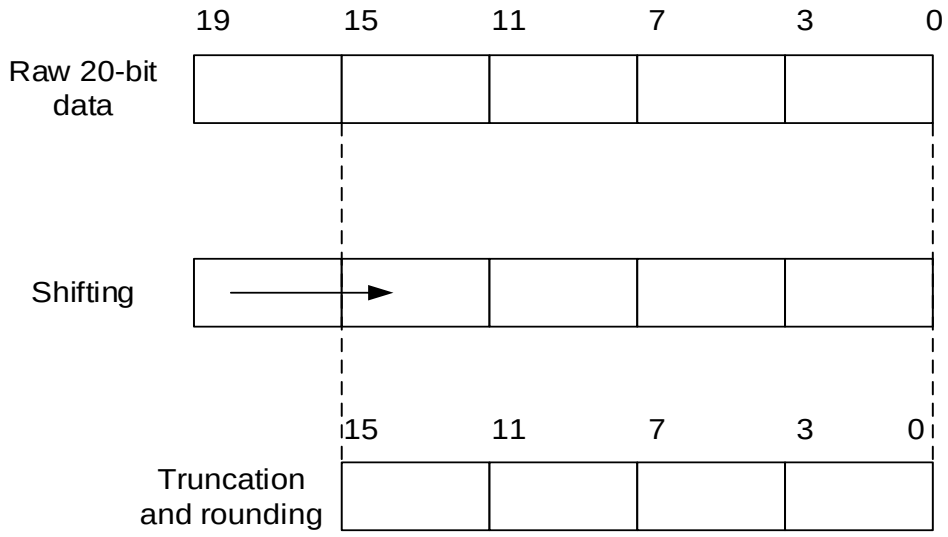
The on-chip hardware oversampling circuit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit. The on-chip hardware oversampling circuit is enabled by OVSEN bit in the ADC_OVSAMPCTL register. It provides a result with the following form, where N and M can be adjusted, and $D_{out}(n)$ is the n-th output digital signal of the ADC:

$$Result=\frac{1}{M}*\sum_{n=0}^{N-1} D_{out}(n) \quad (13\text{-}2)$$

The on-chip hardware oversampling circuit performs the following functions: summing and bit right shifting. The oversampling ratio N is defined by the OVSR[2:0] bits in the ADC_OVSAMPCTL register. It can range from 2x to 256x. The division coefficient M means bit right shifting up to 8-bit. It is configured through the OVSS[3:0] bits in the ADC_OVSAMPCTL register.

Summation units can produce up to 20 bits (256 x 12-bit), which is first shifted right. The upper bits of the result are then truncated, keeping only the 16 least significant bits rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the data register.

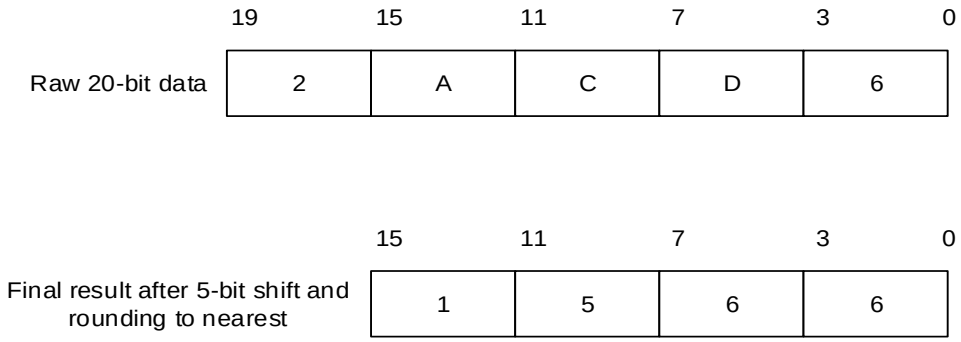
Figure 13-10. 20-bit to 16-bit result truncation



Note: If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

Figure 13-11. Numerical example with 5-bits shift and rounding shows a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 13-11. Numerical example with 5-bits shift and rounding



The *Table 13-6. Maximum output results for N and M combimations (grayed values indicates truncation)* below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFF.

Table 13-6. Maximum output results for N and M combimations (grayed values

indicates truncation)

Oversampling ratio	Max Raw data	No-shift OVSS=0000	1-bit shift OVSS=0001	2-bit shift OVSS=0010	3-bit shift OVSS=0011	4-bit shift OVSS=0100	5-bit shift OVSS=0101	6-bit shift OVSS=0110	7-bit shift OVSS=0111	8-bit shift OVSS=1000
2x	0x1FFE	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF	0x00FF	0x007F	0x003F	0x001F
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF	0x00FF	0x007F	0x003F
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF	0x00FF	0x007F
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF	0x00FF
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

The conversion timings in oversampling mode do not change, and compared to standard conversion mode: the sampling time remains equal throughout the oversampling sequence. New data is supplied every N conversions, and the equivalent delay is equal to:

$$N*t_{ADC}=N*(t_{SMPL}+t_{CONV}) \quad (13\text{-}3)$$

13.5. ADC sync mode

In devices with more than one ADC, the ADC sync mode can be used. In ADC sync mode, the conversion starts alternately or simultaneously triggered by ADC0 to ADC1, according to the sync mode configurated 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 ADC1 must be configured as triggered by the software. However, the external trigger must be enabled for ADC0 and ADC1.

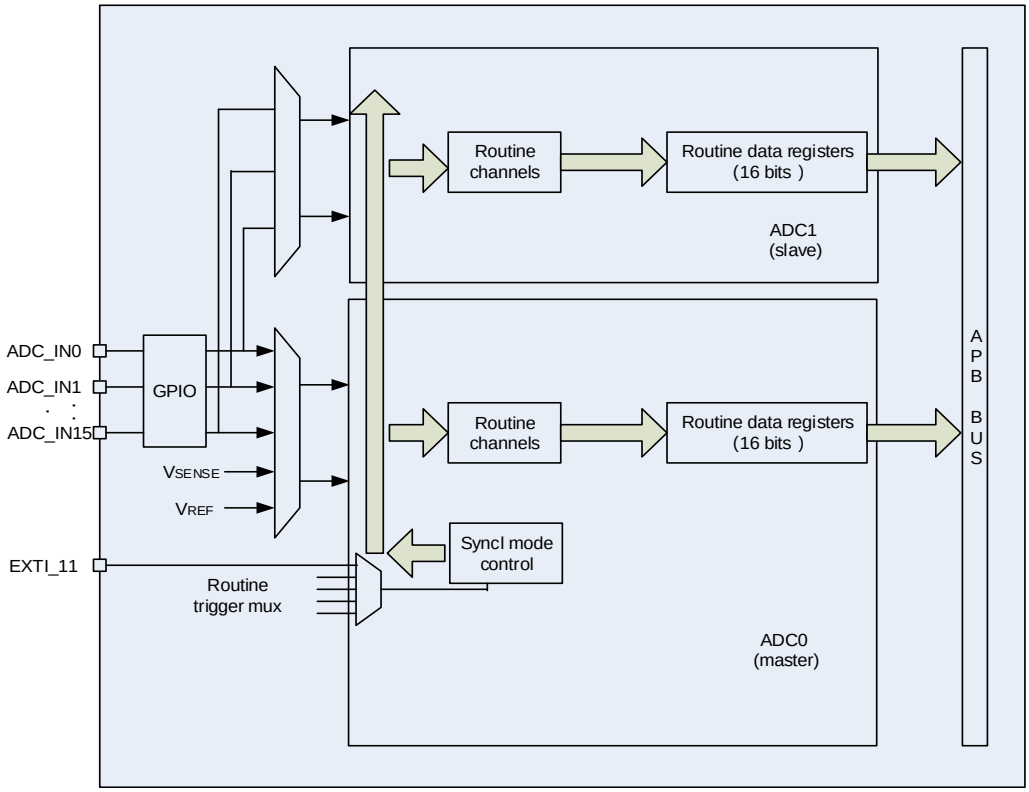
The following modes can be configured in ***Table 13-7. ADC sync mode table***.

Table 13-7. ADC sync mode table

SYNCM[3: 0]	mode
0000	Free mode
0110	Routine parallel mode
0111	Routine follow-up fast mode
1000	Routine follow-up slow mode

In ADC sync mode, the DMA bit must be set even if it is not used; the converted data of ADC1 routine channel can be read from the ADC0 data register.

Figure 13-12. ADC sync block diagram



13.5.1. Free mode

In this mode , each ADC works independently and does not interfere with each other.

13.5.2. Routine parallel mode

This mode converts the routine channel simultaneously. The source of external trigger comes from the ADC0 routine sequence (configured by the ETSRC[2:0] bits in the ADC_CTL1 register) , and ADC1 routine sequence is configured as software trigger mode.

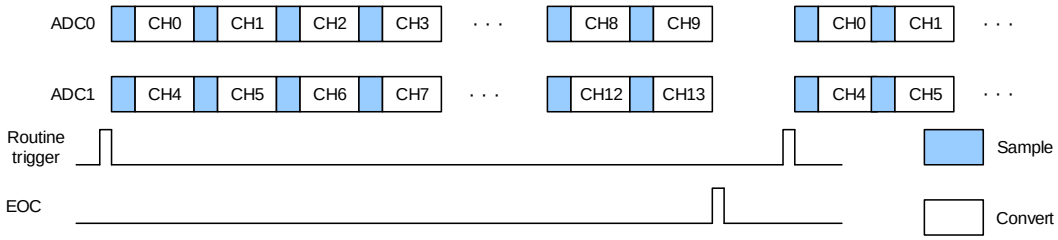
At the end of conversion event on ADC0 or ADC1, an EOC interrupt is generated (if enabled on one of the two ADC interrupt) when the ADC0/ADC1 routine channels are all converted. The behavior of routine parallel mode shows in the ***Figure 13-13. Routine parallel mode on 10 channels***.

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 [31: 16] bits field and the ADC0 converted data in the [15: 0] bits field) to SRAM.

Note:

1. If two ADCs use the same sampling channel, it should be ensured that the channel is not used at the same time.
2. Two channels sampled by two ADCs at the same time should be configured with the same sampling time.

Figure 13-13. Routine parallel mode on 10 channels



13.5.3. Routine follow-up fast mode

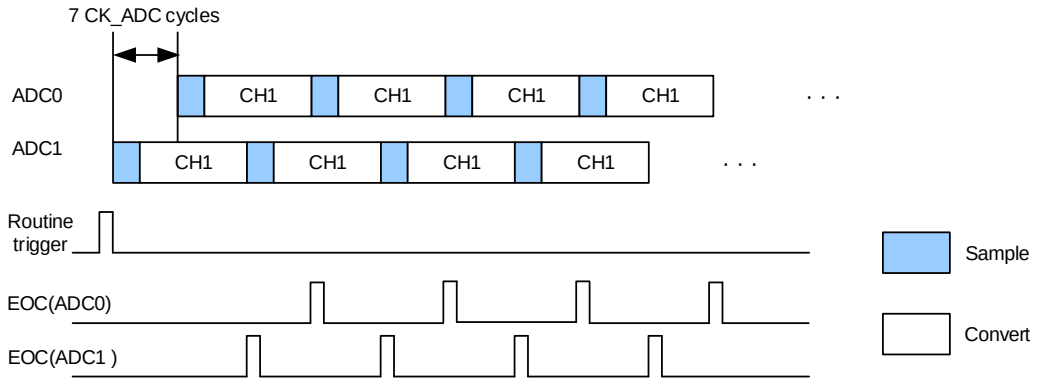
The follow-up fast mode is applicable to sample the same channel of two ADCs. The source of external trigger comes from the ADC0 routine channel (selected by the ETSRC[2:0] bits in the ADC_CTL1 register). When the trigger occurs, ADC1 runs immediately and ADC0 runs after 7 ADC clock cycles.

If the continuous operation mode is enabled for both ADC0 and ADC1, the selected routine channels of two ADCs are continuously converted. The behavior of follow-up fast mode shows in the ***Figure 13-14. Routine follow-up fast mode on routine sequence (the CTN bit of the ADCs are set)***.

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 register containing the ADC1 converted data in the [31: 16] bits field and the ADC0 converted data in the [15: 0] bits field.

Note: The sampling time of the routine channel of the two ADCs should be less than 7 ADC clock cycles.

Figure 13-14. Routine follow-up fast mode on routine sequence (the CTN bit of the ADCs are set)



13.5.4. Routine follow-up slow mode

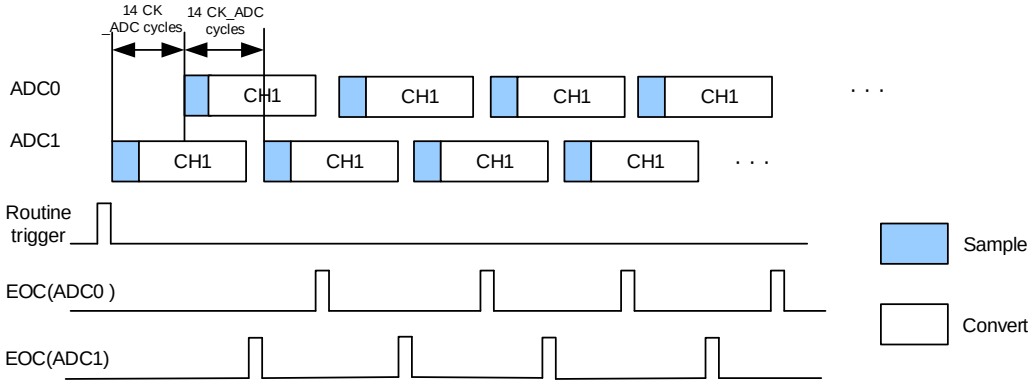
The routine follow-up slow mode is applicable to sample the same channel of two ADCs. The source of external trigger comes from the ADC0 routine channel (selected by the ETSRC[2:0] bits in the ADC_CTL1 register). When the trigger occurs, ADC1 runs immediately, ADC0 runs after 14 ADC clock cycles, after the second 14 ADC clock cycles the ADC1 runs again.

Continuous operation mode can't be used in this mode, because it continuously converts the routine channel. The behavior of follow-up slow mode shows in the ***Figure 13-15. Routine follow-up slow mode on routine sequence channel***.

After an EOC interrupt is generated by ADC0 (if EOCIE bit is set), we can use a 32-bit DMA, which transfers to SRAM the ADC_RDATA register containing the ADC1 converted data in the [31: 16] bits field and the ADC0 converted data in the [15: 0] bits field.

Note: The maximum sampling time allowed is <14 CK_ADC cycles to avoid the overlap between ADC0 and ADC1 sampling phases in the event that they convert the same channel.

Figure 13-15. Routine follow-up slow mode on routine sequence channel



13.6. ADC interrupts

The interrupt can be produced on one of the events:

- End of conversion for routine sequence.
- The analog watchdog event.

The interrupts of ADC0, ADC1 are mapped into the same interrupt vector ISR[18], and ADC2 is mapped into the interrupt vector ISR[47].

13.7. ADC registers

ADC0 base address: 0x4001 2400

ADC1 base address: 0x4001 2800

ADC2 base address: 0x4001 3C00

13.7.1. 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
WDE2	WDE1	Reserved													
rc_w0	rc_w0														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											STRC	Reserved		EOC	WDE0
											rc_w0			rc_w0	rc_w0

Bits	Fields	Descriptions
31	WDE2	Analog watchdog 2 event flag 0: Analog watchdog 2 event is not happened 1: Analog watchdog 2 event is happening Set by hardware when the converted voltage crosses the values programmed in the ADC_WDT2 register. Cleared by software writing 0 to it.
30	WDE1	Analog watchdog 1 event flag 0: Analog watchdog 1 event is not happened 1: Analog watchdog 1 event is happening Set by hardware when the converted voltage crosses the values programmed in the ADC_WDT1 register. Cleared by software writing 0 to it.
29:5	Reserved	Must be kept at reset value.
4	STRC	Start flag of routine sequence 0: Conversion is not started 1: Conversion is started Set by hardware when routine sequence conversion starts. Cleared by software writing 0 to it.
31:5	Reserved	Must be kept at reset value.
1	EOC	End flag of routine sequence conversion 0: No end of routine sequence conversion 1: End ofroutine sequence conversion

		Set by hardware at the end of a routine sequence conversion. Cleared by software writing 0 to it or by reading the ADC_RDATA register.
0	WDE0	Analog watchdog 0 event flag 0: Analog watchdog 0 event is not happened 1: Analog watchdog 0 event is happening Set by hardware when the converted voltage crosses the values programmed in the ADC_WDLT0 and ADC_WDHT0 registers. Cleared by software writing 0 to it.

13.7.2. 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
WDE2IE	WDE1IE	Reserved						RWD0EN	Reserved			SYNCM[3:0]			
rw	rw							rw	rw			rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISNUM[2:0]			Reserved	DISRC	Reserved	WD0SC	SM	Reserved	WDE0IE	EOCIE	WD0CHSEL[4:0]				
rw				rw		rw	rw		rw	rw	rw				

Bits	Fields	Descriptions
31	WDE2IE	Interrupt enable for WDE2 0: Interrupt disable 1: Interrupt enable
30	WDE1IE	Interrupt enable for WDE1 0: Interrupt disable 1: Interrupt enable
29:24	Reserved	Must be kept at reset value.
23	RWD0EN	Routine channel analog watchdog 0 enable 0: Analog watchdog 0 disable 1: Analog watchdog enable
22:20	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. 0110: Routine parallel mode only 0111: Routine follow-up fast mode only 1000: Routine follow-up slow mode only **Note:** 1) These bits are only used in ADC0. 2) Users 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 in routine sequence.
12	Reserved	Must be kept at reset value.
11	DISRC	Discontinuous mode on routine channels 0: Discontinuous operation mode on routine channels disable 1: Discontinuous operation mode on routine channels enable
10	Reserved	Must be kept at reset value.
9	WD0SC	When in scan mode, analog watchdog 0 is effective on a single channel. 0: All channels have analog watchdog function 1: A single channel has analog watchdog function
8	SM	Scan mode 0: scan operation mode disable 1: scan operation mode enable
7	Reserved	Must be kept at reset value.
6	WDE0IE	Interrupt enable for WDE0 0: Interrupt disable 1: Interrupt enable
5	EOCIE	Interrupt enable for EOC 0: Interrupt disable 1: Interrupt enable
4:0	WD0CHSEL[4:0]	Analog watchdog 0 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 Channel16, and Channel17 are internally connected to V_{SSA}.

13.7.3. 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
ETSRC[3]	Reserved							TSVREN	SWRCST	Reserved	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
Reserved				DAL	Reserved.		DMA	Reserved	CALNUM[2:0]			RSTCLB	CLB	CTN	ADCON
				rw			rw		rw			rw	rw	rw	rw

Bits	Fields	Descriptions
31	ETSRC[3]	External trigger select for routine channel, refer to ETSRC[2:0] description.
30:24	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	Software start conversion of routine sequence . Set 1 on this bit starts a conversion of a routine sequence if ETSRC is 111. It is set by software and cleared by software or by hardware immediately after the conversion starts.
21	Reserved	Must be kept at reset value.
20	ETERC	External trigger enable for routine sequence 0: External trigger for routine sequence disable 1: External trigger for routine sequence enable
19:17	ETSRC[2:0]	External trigger select for routine sequence For ADC0 and ADC1: 0000: TIMER0_CH0 0001: TIMER0_CH1 0010: TIMER0_CH2

		0011: TIMER1_CH1 0100: TIMER2_TRGO 0101: TIMER3_CH3 0110: EXTI Line11/ TIMER7_TRGO 0111: SWRCST 1000: SHRTIMER_ADCTRG0 1001: SHRTIMER_ADCTRG2 Others: Reserved For ADC2: 0000: TIMER2_CH0 0001: TIMER1_CH2 0010: TIMER0_CH2 0011: TIMER7_CH0 0100: TIMER7_TRGO 0101: TIMER4_CH0 0110: TIMER4_CH2 0111: SWRCST 1xxx: Reserved
16:12	Reserved	Must be kept at reset value.
11	DAL	Data storage alignment mode 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	Reserved	Must be kept at reset value.
6:4	CALNUM[2:0]	Calibration Times These bits define the calibration times for ADC. 000: 1 time 001: 2 times 010: 4 times 011: 8 times 100: 16 times 101: 32 times Others: reserved.
3	RSTCLB	Reset calibration register 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 operation mode disable 1: Continuous operation 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

13.7.4. 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		

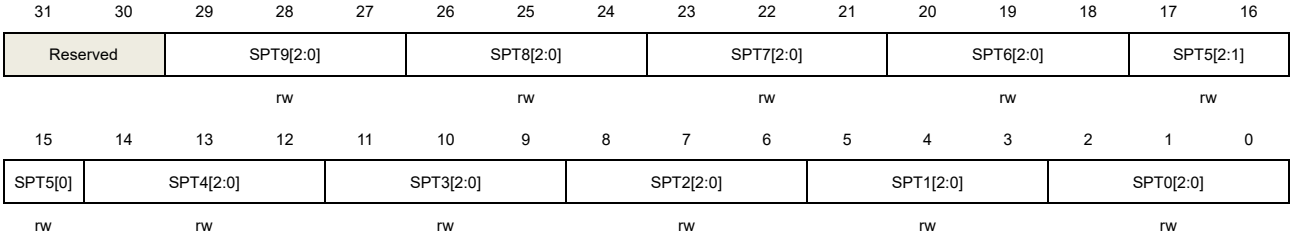
Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23:21	SPT17[2:0]	refer to SPT10[2:0] description
20:18	SPT16[2:0]	refer to SPT10[2:0] description
17:15	SPT15[2:0]	refer to SPT10[2:0] description
14:12	SPT14[2:0]	refer to SPT10[2:0] description
11:9	SPT13[2:0]	refer to SPT10[2:0] 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: Channel sampling time is 1.5 cycles 001: Channel sampling time is 7.5 cycles

010: Channel sampling time is 13.5 cycles
011: Channel sampling time is 28.5 cycles
100: Channel sampling time is 41.5 cycles
101: Channel sampling time is 55.5 cycles
110: Channel sampling time is 71.5 cycles
111: Channel sampling time is 239.5 cycles

13.7.5. Sample time register 1 (ADC_SAMPT1)

Address offset: 0x10
Reset value: 0x0000 0000

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



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: Channel sampling time is 1.5 cycles 001: Channel sampling time is 7.5 cycles 010: Channel sampling time is 13.5 cycles 011: Channel sampling time is 28.5 cycles 100: Channel sampling time is 41.5 cycles 101: Channel sampling time is 55.5 cycles

110: Channel sampling time is 71.5 cycles

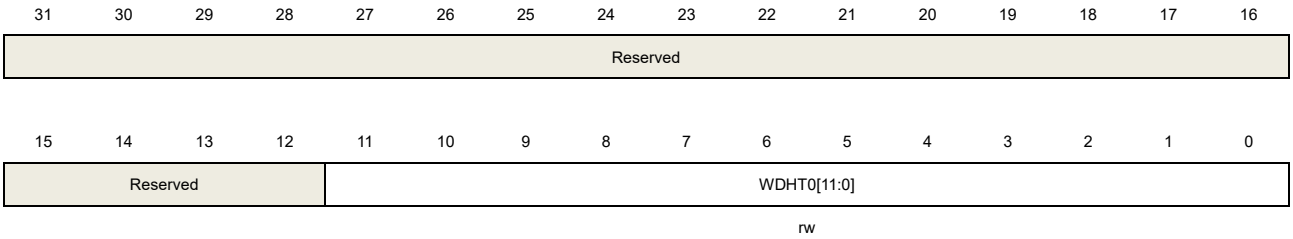
111: Channel sampling time is 239.5 cycles

13.7.6. Watchdog high threshold register 0 (ADC_WDHT0)

Address offset: 0x24

Reset value: 0x0000 0FFF

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



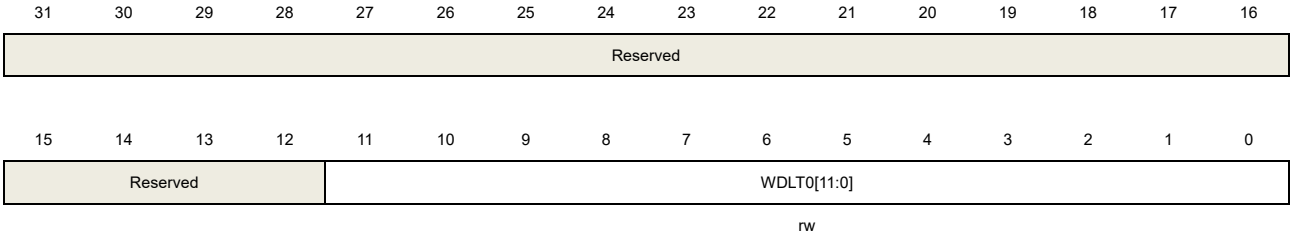
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	WDHT0[11:0]	High threshold for analog watchdog 0 These bits define the high threshold for the analog watchdog 0.

13.7.7. Watchdog low threshold register 0 (ADC_WDLT0)

Address offset: 0x28

Reset value: 0x0000 0000

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



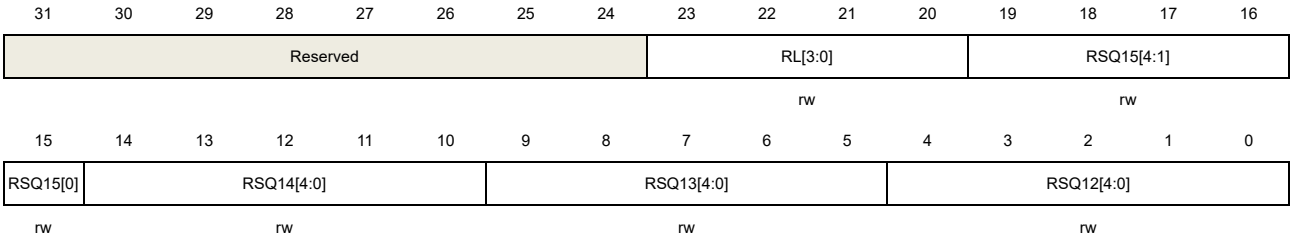
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	WDLT0[11:0]	Low threshold for analog watchdog 0 These bits define the low threshold for the analog watchdog 0.

13.7.8. Routine sequence register 0 (ADC_RSQ0)

Address offset: 0x2C

Reset value: 0x0000 0000

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



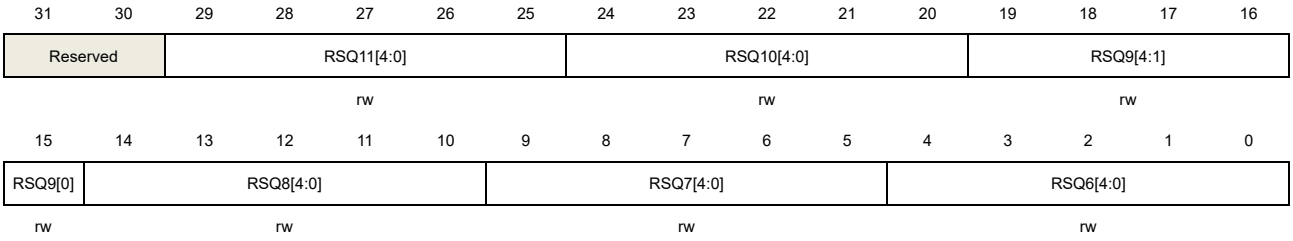
Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23:20	RL[3:0]	Routine sequence length. The total number of conversion in routine sequence equals to RL[3:0]+1.
19:15	RSQ15[4:0]	refer to RSQ0[4:0] description
14:10	RSQ14[4:0]	refer to RSQ0[4:0] description
9:5	RSQ13[4:0]	refer to RSQ0[4:0] description
4:0	RSQ12[4:0]	refer to RSQ0[4:0] description

13.7.9. Routine sequence register 1 (ADC_RSQ1)

Address offset: 0x30

Reset value: 0x0000 0000

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



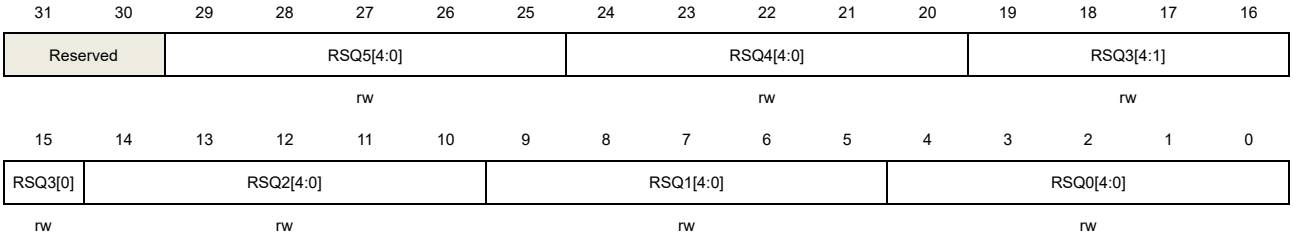
Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:25	RSQ11[4:0]	refer to RSQ0[4:0] description
24:20	RSQ10[4:0]	refer to RSQ0[4:0] description
19:15	RSQ9[4:0]	refer to RSQ0[4:0] description
14:10	RSQ8[4:0]	refer to RSQ0[4:0] description

9:5	RSQ7[4:0]	refer to RSQ0[4:0] description
4:0	RSQ6[4:0]	refer to RSQ0[4:0] description

13.7.10. Routine sequence register 2 (ADC_RSQ2)

Address offset: 0x34
Reset value: 0x0000 0000

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

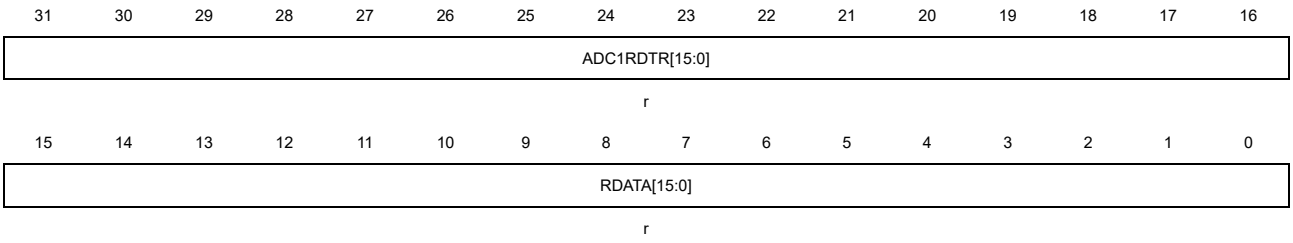


Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:25	RSQ5[4:0]	refer to RSQ0[4:0] description
24:20	RSQ4[4:0]	refer to RSQ0[4:0] description
19:15	RSQ3[4:0]	refer to RSQ0[4:0] description
14:10	RSQ2[4:0]	refer to RSQ0[4:0] description
9:5	RSQ1[4:0]	refer to RSQ0[4:0] description
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 routine sequence.

13.7.11. Routine data register (ADC_RDATA)

Address offset: 0x4C
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions

31:16	ADC1RDTR[15:0]	ADC1 routine channel data In ADC0: In sync mode, these bits contain the routine data of ADC1. These bits are only used in ADC0.
15:0	RDATA[15:0]	Routine channel data These bits contain the conversion result from routine channel, which is read only.

13.7.12. Oversample control register (ADC_OVSAMPCTL)

Address offset: 0x80

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		DRES[1:0]		Reserved		TOVS	OVSS[3:0]				OVSR[2:0]			Reserved	OVSEN
		rw				rw	rw				rw				rw

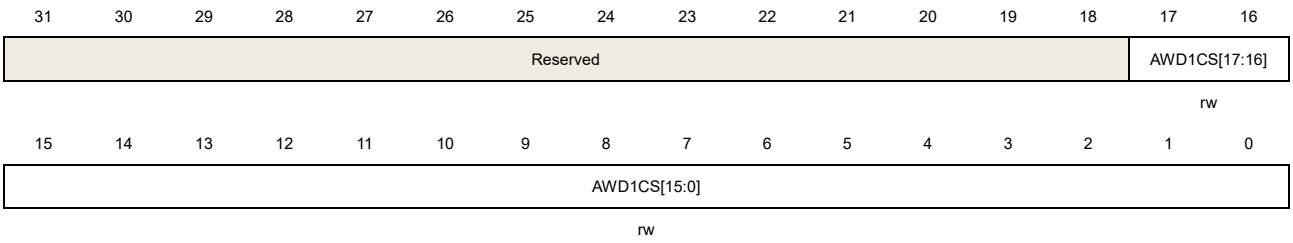
Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:12	DRES[1:0]	ADC resolution 00: 12bit 01: 10bit 10: 8bit 11: 6bit
11:10	Reserved	Must be kept at reset value.
9	TOVS	Triggered Oversampling This bit is set and cleared by software. 0: All oversampled conversions for a channel are done consecutively after a trigger. 1: Each conversion needs a trigger for an oversampled channel and the number of triggers is determined by the oversampling ratio(OVSR[2:0]). **Note:** The software allows this bit to be written only when ADCON=0 (this ensures that no conversion is in progress).
8:5	OVSS[3:0]	Oversampling shift This bit is set and cleared by software. 0000: No shift 0001: Shift 1-bit 0010: Shift 2-bits 0011: Shift 3-bits 0100: Shift 4-bits

		0101: Shift 5-bits 0110: Shift 6-bits 0111: Shift 7-bits 1000: Shift 8-bits Other values are reserved. **Note:** The software allows this bit to be written only when ADCON=0 (this ensures that no conversion is in progress).
4:2	OVSR[2:0]	Oversampling ratio This bit filed defines the number of oversampling ratio. 000: 2x 001: 4x 010: 8x 011: 16x 100: 32x 101: 64x 110: 128x 111: 256x **Note:** The software allows this bit to be written only when ADCON=0 (this ensures that no conversion is in progress).
1	Reserved	Must be kept at reset value.
0	OVSEN	Oversampler Enable This bit is set and cleared by software. 0: Oversampler disabled 1: Oversampler enabled **Note:** The software allows this bit to be written only when ADCON=0 (this ensures that no conversion is in progress).

13.7.13. Watchdog 1 channel selection register (ADC_WD1SR)

Address offset: 0xA0
Reset value: 0x00000000

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



Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.

17:0	AWD1CS[17:0]	Analog watchdog 1 channel selection These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 1. AWD1CS[n] = 0: ADC analog input channel n is not monitored by analog watchdog 1 AWD1CS[n] = 1: ADC analog input channel n is monitored by analog watchdog 1 When AWD1CH[17:0] = 000..0, the analog watchdog 1 is disabled Note: 1) The channels selected by AWD1CS must be also selected into the ADC_RSQn or ADC_ISQ registers. 2) Software is allowed to write these bits only when the ADC is disabled (ADCON =0).

13.7.14. Watchdog 2 channel selection register (ADC_WD2SR)

Address offset: 0xA4
Reset value: 0x00000000

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														AWD2CS[17:16]	
														rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CS[15:0]															
rw															

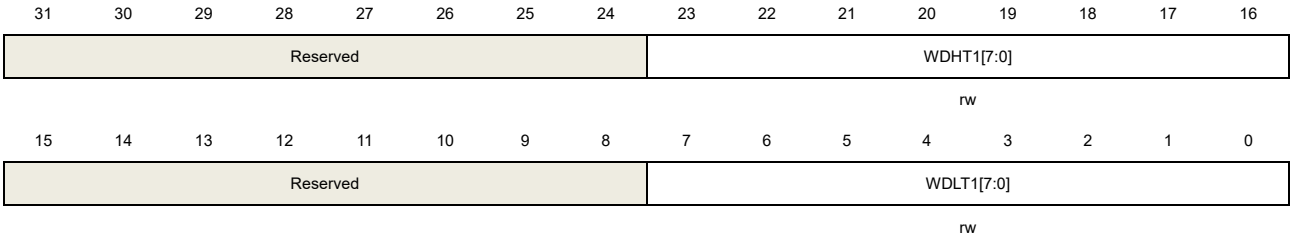
Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.
17:0	AWD2CS[17:0]	Analog watchdog 2 channel selection These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2. AWD2CS[n] = 0: ADC analog input channel n is not monitored by analog watchdog 2 AWD2CS[n] = 1: ADC analog input channel n is monitored by analog watchdog 2 When AWD2CH[17:0] = 000..0, the analog watchdog 2 is disabled Note: The channels selected by AWD2CS must be also selected into the ADC_RSQn or ADC_ISQ registers. Note: Software is allowed to write these bits only when the ADC is disabled (ADCON =0).

13.7.15. Watchdog threshold register 1 (ADC_WDT1)

Address offset: 0xA8

Reset value: 0x00FF 0000

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



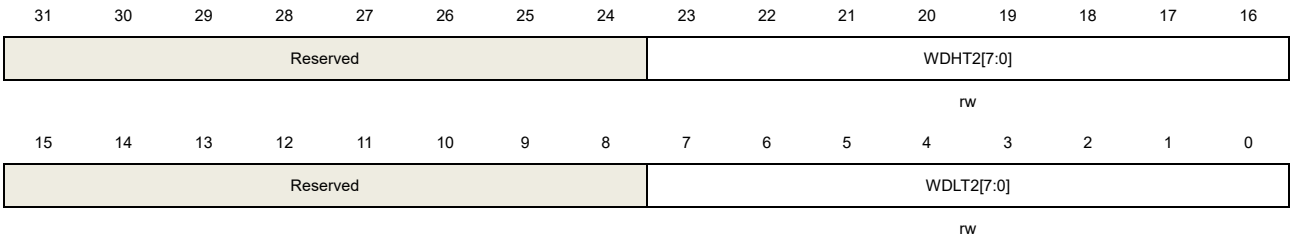
Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23:16	WDHT1[7:0]	High threshold for analog watchdog 1 These bits define the high threshold for the analog watchdog 1. **Note:** Software is allowed to write these bits only when the ADC is disabled (ADCON =0).
15:8	Reserved	Must be kept at reset value.
7:0	WDLT1[7:0]	Low threshold for analog watchdog 1 These bits define the high threshold for the analog watchdog 1. Note: Software is allowed to write these bits only when the ADC is disabled (ADCON =0).

13.7.16. Watchdog threshold register 2 (ADC_WDT2)

Address offset: 0xAC

Reset value: 0x00FF 0000

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



Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23:16	WDHT2[7:0]	High threshold for analog watchdog 2 These bits define the high threshold for the analog watchdog 2. Note: Software is allowed to write these bits only when the ADC is disabled (ADCON =0).

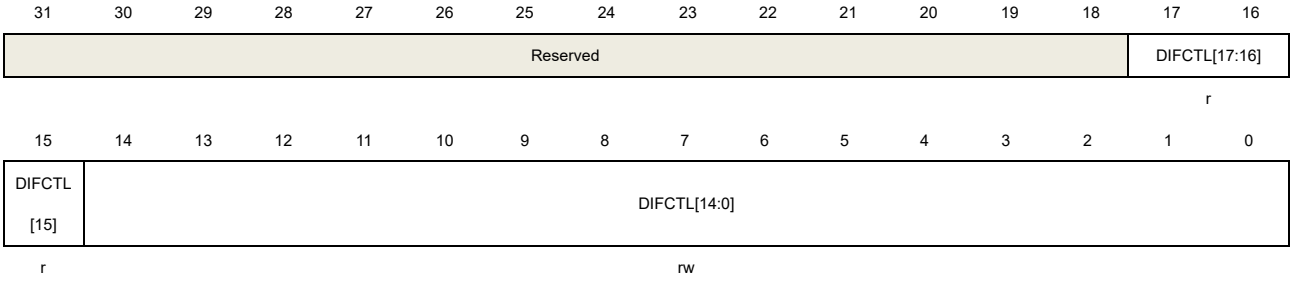
15:8	Reserved	Must be kept at reset value.
7:0	WDLT2[7:0]	Low threshold for analog watchdog 2 These bits define the high threshold for the analog watchdog 2. **Note:** Software is allowed to write these bits only when the ADC is disabled (ADCON =0).

13.7.17. Differential mode control register (ADC_DIFCTL)

Address offset: 0xB0

Reset value: 0x00000000

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



Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.
17:15	DIFCTL[17:15]	Differential mode for channel 17..15. These bits are read only. These channels are forced to single-ended input mode (either connected to a single-ended I/O port or to an internal channel).
14:0	DIFCTL[14:0]	Differential mode for channel 14..0. These bits are configured to select whether a channel is in single-ended or differential mode. DIFCTL[i] = 0: ADC analog input channel-i is configured in single-ended mode DIFCTL[i] = 1: ADC analog input channel-i is configured in differential mode **Note:** Software is allowed to write these bits only when the ADC is disabled (ADCON =0).

14. Digital-to-analog converter (DAC)

14.1. Overview

The Digital-to-analog converter converts 12-bit digital data to a voltage on the external pins. The digital data can be configured to 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 DAC channels can work independently or concurrently.

14.2. Characteristics

The main features of DAC are as follows:

- 8-bit or 12-bit resolution.
- Left or right data alignment.
- DMA capability for each channel and underrun function.
- Conversion update synchronously.
- Conversion triggered by external triggers.
- Configurable internal buffer.
- Extern voltage reference, V_{REFP}.
- Output FIFO.
- Noise wave generation (LFSR noise mode and triangle noise mode).
- Two DAC channels in concurrent mode.

Figure 14-1. DAC block diagram and *Table 14-1. DAC I/O description* show the block diagram of DAC and the pin description of DAC, respectively.

Figure 14-1. DAC block diagram

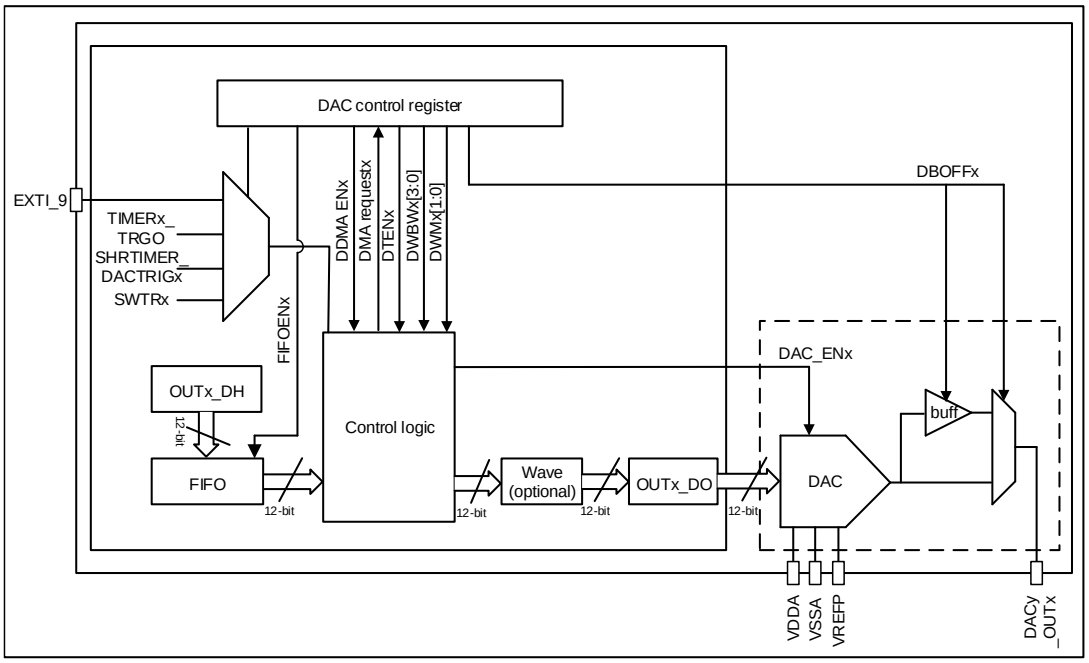


Table 14-1. DAC I/O description

Name	Description	Signal type
V_{DDA}	Analog power supply	Input, analog supply
V_{SSA}	Ground for analog power supply	Input, analog supply ground
V_{REFP}	Positive reference voltage of DAC	Input, analog positive reference
DACy_OUTx	DAC analog output	Analog output signal

The below table details the triggers and outputs of the DAC.

Table 14-2. DAC triggers and outputs summary

	DAC0	
Channel	**Channel0**	**Channel1**
DAC outputs connected to I / Os	PA4	PA5
DAC output buffer	•	•
DAC software trigger	•	
DAC trigger signals from EXTI	EXTI_9	
DAC trigger signals from TIMER	TIMER1_TRGO TIMER2_TRGO in connectivity line devices TIMER7_TRGO in other type devices TIMER3_TRGO TIMER4_TRGO TIMER5_TRGO TIMER6_TRGO	
DAC trigger	SHRTIMER_DACTRIG0	

	DAC0
signals from SHRTIMER	SHRTIMER_DACTRIG1 SHRTIMER_DACTRIG2

Note: The GPIO pins should be configured to analog mode before enable the DAC module.

14.3. Function overview

14.3.1. DAC enable

The DAC can be turned on by setting the DENx bit in the DAC_CTL0 register. A t_{WAKEUP} time is needed to startup the analog DAC submodule.

14.3.2. DAC output buffer

For 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 bit in the DAC_CTL0 register.

14.3.3. DAC data configuration

The 12-bit DAC holding data (OUTx_DH) can be configured by writing any one of the DAC_OUTx_R12DH, DAC_OUTx_L12DH and DAC_OUTx_R8DH registers. When the data is loaded by DAC_OUTx_R8DH register, only the MSB 8 bits are configurable, the LSB 4 bits are forced to 4'b0000.

14.3.4. DAC trigger

The DAC conversion can be triggered by software or rising edge of external trigger source. The DAC external trigger is enabled by setting the DTENx bits in the DAC_CTL0 register. The DAC external triggers are selected by the DTSELx bits in the DAC_CTL0 register, which is shown as ***Table 14-3. Triggers of DAC***.

Table 14-3. Triggers of DAC

DTSELx[3:0]	Trigger Source	Trigger Type
4b'0000	TIMER5_TRGO	Hardware trigger
4b'0001	TIMER2_TRGO in connectivity line devices; TIMER7_TRGO in other type devices	
4b'0010	TIMER6_TRGO	
4b'0011	TIMER4_TRGO	

DTSELx[3:0]	Trigger Source	Trigger Type
4b'0100	TIMER1_TRGO	
4b'0101	TIMER3_TRGO	
4b'0110	EXTI_9	
4b'0111	SWTR	Software trigger
4b'1000	SHRTIMER_DACTRIG0	Hardware trigger
4b'1001	SHRTIMER_DACTRIG1	
4b'1010	SHRTIMER_DACTRIG2	
4b'1011~1111	Reserved	Reserved

The TIMERx_TRGO signals are generated from the timers, the SHRTIMER_DACTRIGx signals are generated from the SHRTIMER, while the software trigger can be generated by setting the SWTRx bits in the DAC_SWT register.

14.3.5. DAC conversion

If the external trigger is enabled by setting the DTENx bit in DAC_CTL0 register, the DAC holding data is transferred to the DAC output data (DAC_OUTx_DO) register when the selected trigger event happened. When the external trigger is disabled, the transfer is performed automatically.

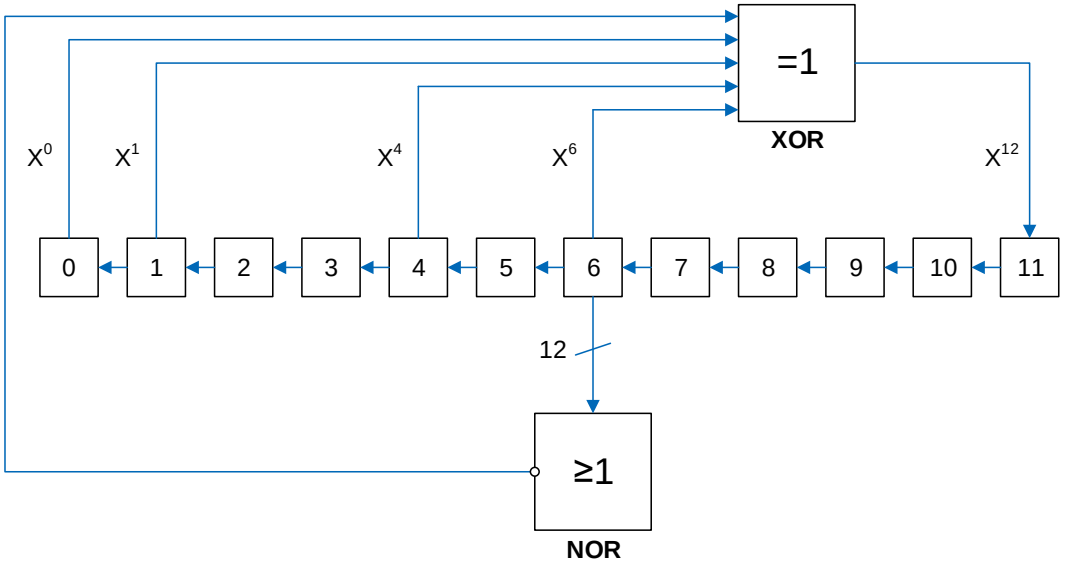
When the DAC holding data (OUTx_DH) is loaded into the DAC_OUTx_DO register, after the time $t_{SETTLING}$ which is determined by the analog output load and the power supply voltage, the analog output is valid.

14.3.6. DAC noise wave

There are two methods of adding noise wave to the DAC output data: LFSR noise wave mode and Triangle wave mode. The noise wave mode can be selected by the DWMx bits in the DAC_CTL0 register. The amplitude of the noise can be configured by the DAC noise wave bit width (DWBWx) bits in the DAC_CTL0 register.

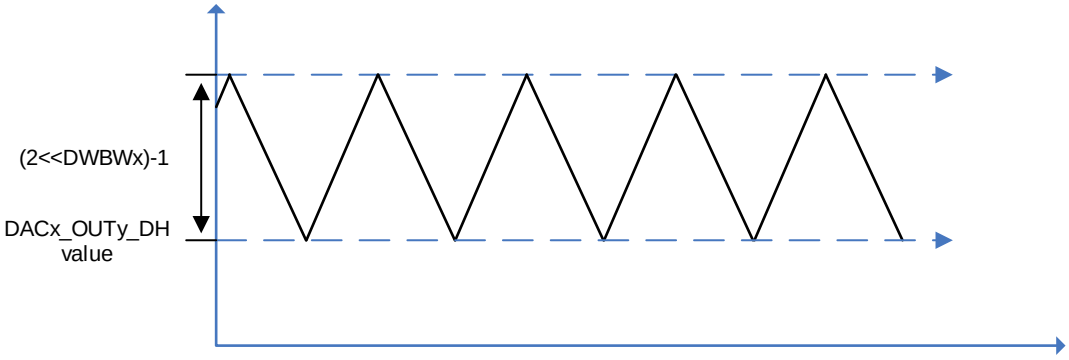
LFSR noise wave mode: there is a Linear Feedback Shift Register (LFSR) in the DAC control logic, it controls the LFSR noise signal which is added to the OUTx_DH value, and then the result is stored into the DAC_OUTx_DO register. When the configured DAC noise wave bit width is less than 12, the noise signal equals to the LSB DWBWx bits of the LFSR register, while the MSB bits are masked.

Figure 14-2. DAC LFSR algorithm



Triangle noise mode: a triangle signal is added to the OUTx_DH value, and then the result is stored into the DAC_OUTx_DO register. The minimum value of the triangle signal is 0, while the maximum value of the triangle signal is (2 << DWBWx) - 1.

Figure 14-3. DAC triangle noise wave



14.3.7. DAC output voltage

The following equation determines the analog output voltage on the DAC pin.

$$V_{DAC\_OUT}=V_{REFP}*OUTx\_DO/4096 \quad (14\text{-}1)$$

The digital input is linearly converted to an analog output voltage and its range is 0 to V_{REFP}.

14.3.8. DMA request

When the external trigger is enabled, the DMA request is enabled by setting the DDMAENx bit of the DAC_CTL0 register. A DMA request will be generated when an external hardware trigger (not a software trigger) occurs.

If the second external trigger arrives before confirming the previous request, the new request will not be serviced, and an underrun error event occurs. The DDUDRx bit in the DAC_STAT0 register is set, an interrupt will be generated if the DDUDRIEx bit in the DAC_CTL0 register is set. The DMA request will be stalled until the DDUDRx bit is cleared.

14.3.9. DAC concurrent conversion

When the two output channels work at the same time, for maximum bus bandwidth utilization in specific applications, two output channels can be configured in concurrent mode. In concurrent mode, the OUTx_DH and OUTx_DO value will be updated at the same time.

There are three concurrent registers that can be used to load the OUTx_DH value: DACC_R8DH, DACC_R12DH and DACC_L12DH. User just need to access a unique register to realize driving two DAC channels at the same time.

When external trigger is enabled, please ensure both DTENx bits be set, DTSEL0/DTSEL1 bits be same to guarantee the simultaneous trigger.

When DMA is enabled, please ensure any DDMAENx bit in one DAC be set.

The noise mode and noise bit width can be configured either the same or different, depending on the application scenario.

14.3.10. DAC output FIFO

There is a 4-depth data FIFO which is implemented between the data hold register and the output register. The FIFO can be enabled by setting the FIFOENx bit in the DAC_CTL1 register.

The DDMAENx bit should be reset (DAC_OUTx DMA mode is disabled) and DTENx bit should be set (DAC_OUTx external trigger is enabled), when FIFOENx is set.

When the data FIFO is full, the FIFO is in full status, and the FIFOFx bit will be set to 1. When the data FIFO is empty, the FIFO is in empty status, and the FIFOEx bit will be set to 1.

When the data FIFO is full while no trigger comes, the overflow flag FIFOOVRx bit will be set to 1. When the data FIFO is empty while a trigger is coming, the underflow flag FIFOUDRx bit is set to 1. Interrupts will be generated when an overflow or underflow condition occurs if the corresponding interrupt enable bit is set.

The status (full, empty, overflow or underflow) of the data FIFO can be gotten by reading the DAC_STAT1 register.

14.4. Register definition

DAC0 base address: 0x4000 7400

14.4.1. DACx control register 0 (DAC_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	DTSEL1[3]	DDUDRIE1	DDMAEN1	DWBW1[3:0]				DWM1[1:0]		DTSEL1[2:0]			DTEN1	DBOFF1	DEN1
	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
Reserved	DTSEL0[3]	DDUDRIE0	DDMAEN0	DWBW0[3:0]				DWM0[1:0]		DTSEL0[2:0]			DTEN0	DBOFF0	DEN0
	rw	rw	rw	rw				rw		rw			rw	rw	rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30	DTSEL1[3]	DACx_OUT1 trigger selection bit[3], refer to DTSEL1[2:0]
29	DDUDRIE1	DACx_OUT1 DMA underrun interrupt enable 0: DACx_OUT1 DMA underrun interrupt disabled 1: DACx_OUT1 DMA underrun interrupt enabled
28	DDMAEN1	DACx_OUT1 DMA enable 0: DACx_OUT1 DMA mode disabled 1: DACx_OUT1 DMA mode enabled
27:24	DWBW1[3:0]	DACx_OUT1 noise wave bit width These bits specify bit width of the noise wave signal of DACx_OUT1. 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]	DACx_OUT1 noise wave mode

		These bits specify the mode selection of the noise wave signal of DACx_OUT1 when external trigger of DACx_OUT1 is enabled (DTEN1=1). 00: wave disabled 01: LFSR noise mode 1x: Triangle noise mode
21:19	DTSEL1[2:0]	DACx_OUT1 trigger selection These bits are combined with DTSEL1[3] to select the external event used to trigger DAC and only used if bit DTEN1 = 1. 0000: TIMER5 TRGO 0001: TIMER2 TRGO (connectivity line devices); TIMER7 TRGO (other type devices) 0010: TIMER6 TRGO 0011: TIMER4 TRGO 0100: TIMER1 TRGO 0101: TIMER3 TRGO 0110: EXTI line 9 0111: Software trigger 1000: SHRTIMER_DACTRIG0 1001: SHRTIMER_DACTRIG1 1010: SHRTIMER_DACTRIG2 1011~1111: Reserved
18	DTEN1	DACx_OUT1 trigger enable 0: DACx_OUT1 trigger disabled 1: DACx_OUT1 trigger enabled
17	DBOFF1	DACx_OUT1 output buffer turn off 0: DACx_OUT1 output buffer turns on to reduce the output impedance and improve the driving capability 1: DACx_OUT1 output buffer turns off
16	DEN1	DACx_OUT1 enable 0: DACx_OUT1 disabled 1: DACx_OUT1 enabled
15	Reserved	Must be kept at reset value
14	DTSEL0[3]	DACx_OUT0 trigger selection bit 3, refer to DTSEL0[2:0]
13	DDUDRIE0	DACx_OUT0 DMA underrun interrupt enable 0: DACx_OUT0 DMA underrun interrupt disabled 1: DACx_OUT0 DMA underrun interrupt enabled
12	DDMAEN0	DACx_OUT0 DMA enable 0: DACx_OUT0 DMA mode disabled 1: DACx_OUT0 DMA mode enabled

11:8	DWBW0[3:0]	DACx_OUT0 noise wave bit width These bits specify bit width of the noise wave signal of DACx_OUT0. 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]	DACx_OUT0 noise wave mode These bits specify the mode selection of the noise wave signal of DACx_OUT0 when external trigger of DACx_OUT0 is enabled (DTEN0=1). 00: Wave disabled 01: LFSR noise mode 1x: Triangle noise mode
5:3	DTSEL0[2:0]	DACx_OUT0 trigger selection These bits are combined with DTSEL0[3] to select the external event used to trigger DAC and only used if bit DTEN0 = 1. 0000: TIMER5 TRGO 0001: TIMER2 TRGO (connectivity line devices); TIMER7 TRGO (other type devices) 0010: TIMER6 TRGO 0011: TIMER4 TRGO 0100: TIMER1 TRGO 0101: TIMER3 TRGO 0110: EXTI line 9 0111: Software trigger 1000: SHRTIMER_DACTRIG0 1001: SHRTIMER_DACTRIG1 1010: SHRTIMER_DACTRIG2 1011~1111: Reserved
2	DTEN0	DACx_OUT0 trigger enable 0: DACx_OUT0 trigger disabled 1: DACx_OUT0 trigger enabled
1	DBOFF0	DACx_OUT0 output buffer turn off

		0: DACx_OUT0 output buffer turns on to reduce the output impedance and improve the driving capability 1: DACx_OUT0 output buffer turns off
0	DEN0	DACx_OUT0 enable 0: DACx_OUT0 disabled 1: DACx_OUT0 enabled

14.4.2. DACx 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	DACx_OUT1 software trigger, cleared by hardware. 0: Software trigger disabled 1: Software trigger enabled
0	SWTR0	DACx_OUT0 software trigger, cleared by hardware. 0: Software trigger disabled 1: Software trigger enabled

14.4.3. DACx_OUT0 12-bit right-aligned data holding register (DAC_OUT0_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				OUT0_DH[11:0]											
				rw											

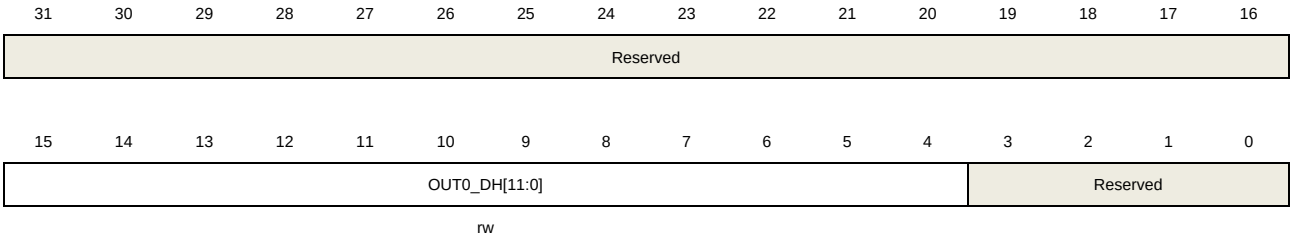
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	OUT0_DH[11:0]	DACx_OUT0 12-bit right-aligned data. These bits specify the data that is to be converted by DACx_OUT0.

14.4.4. DACx_OUT0 12-bit left-aligned data holding register (DAC_OUT0_L12DH)

Address offset: 0x0C

Reset value: 0x0000 0000

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



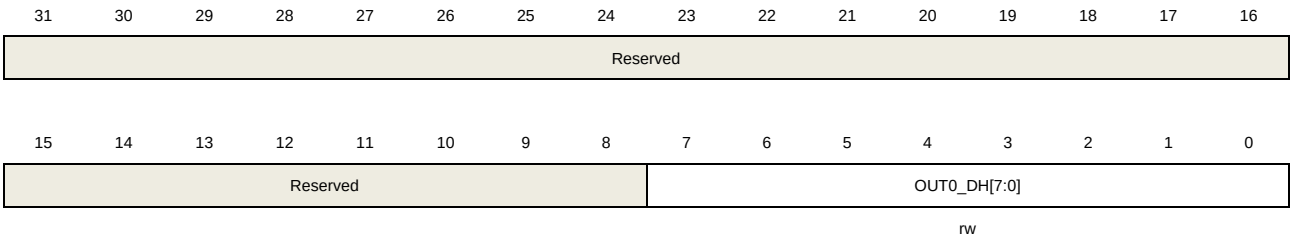
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:4	OUT0_DH[11:0]	DACx_OUT0 12-bit left-aligned data. These bits specify the data that is to be converted by DACx_OUT0.
3:0	Reserved	Must be kept at reset value.

14.4.5. DACx_OUT0 8-bit right-aligned data holding register (DAC_OUT0_R8DH)

Address offset: 0x10

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.

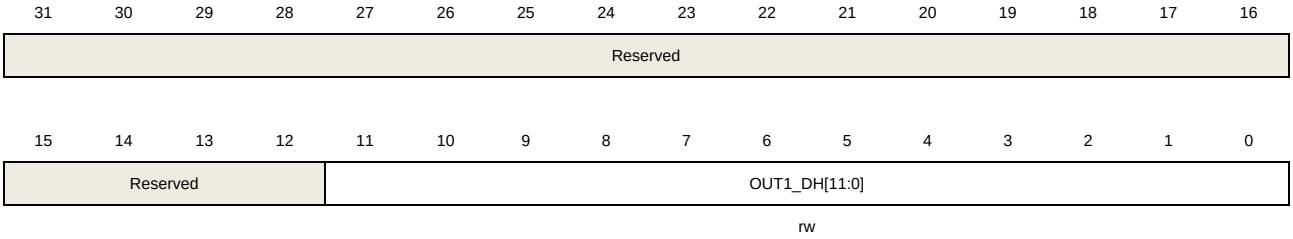
7:0	OUT0_DH[7:0]	DACx_OUT0 8-bit right-aligned data. These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT0.

14.4.6. DACx_OUT1 12-bit right-aligned data holding register (DAC_OUT1_R12DH)

Address offset: 0x14

Reset value: 0x0000 0000

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



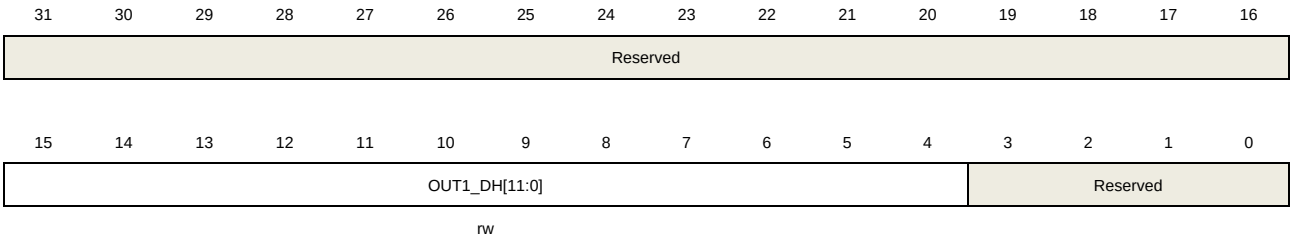
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	OUT1_DH[11:0]	DACx_OUT1 12-bit right-aligned data. These bits specify the data that is to be converted by DACx_OUT1.

14.4.7. DACx_OUT1 12-bit left-aligned data holding register (DAC_OUT1_L12DH)

Address offset: 0x18

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:4	OUT1_DH[11:0]	DACx_OUT1 12-bit left-aligned data. These bits specify the data that is to be converted by DACx_OUT1.

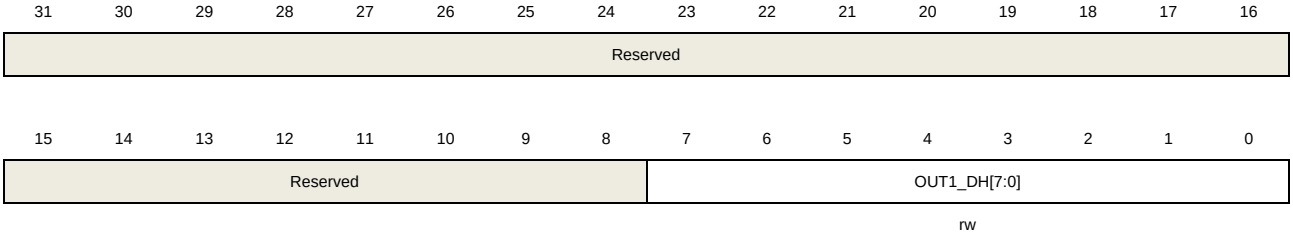
3:0	Reserved	Must be kept at reset value.

14.4.8. DACx_OUT1 8-bit right-aligned data holding register (DAC_OUT1_R8DH)

Address offset: 0x1C

Reset value: 0x0000 0000

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



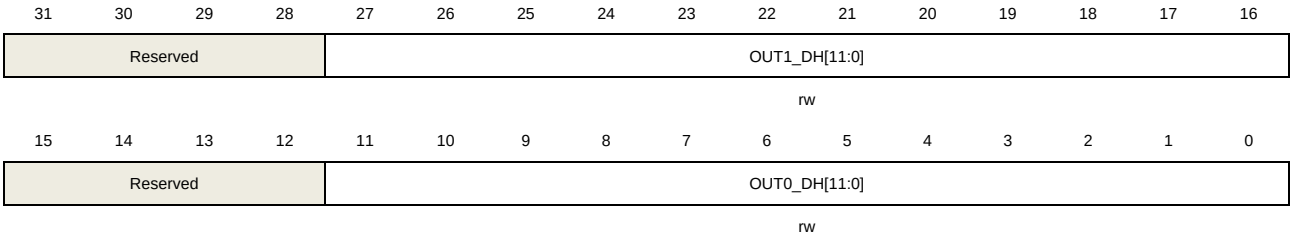
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	OUT1_DH[7:0]	DACx_OUT1 8-bit right-aligned data These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT1.

14.4.9. DACx 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).

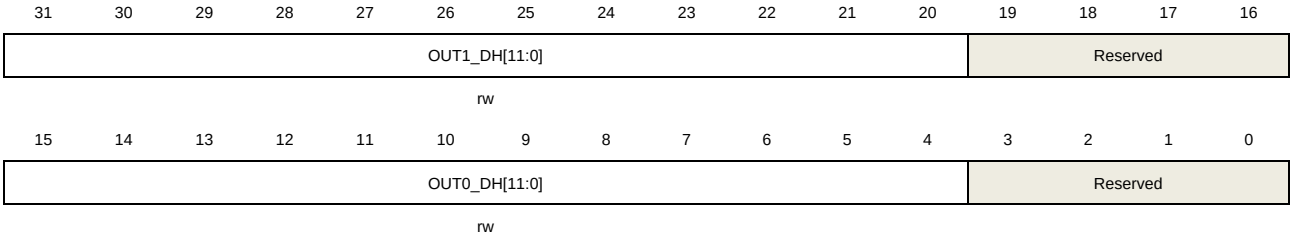


Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:16	OUT1_DH[11:0]	DACx_OUT1 12-bit right-aligned data These bits specify the data that is to be converted by DACx_OUT1.
15:12	Reserved	Must be kept at reset value.
11:0	OUT0_DH[11:0]	DACx_OUT0 12-bit right-aligned data These bits specify the data that is to be converted by DACx_OUT0.

14.4.10. DACx 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)

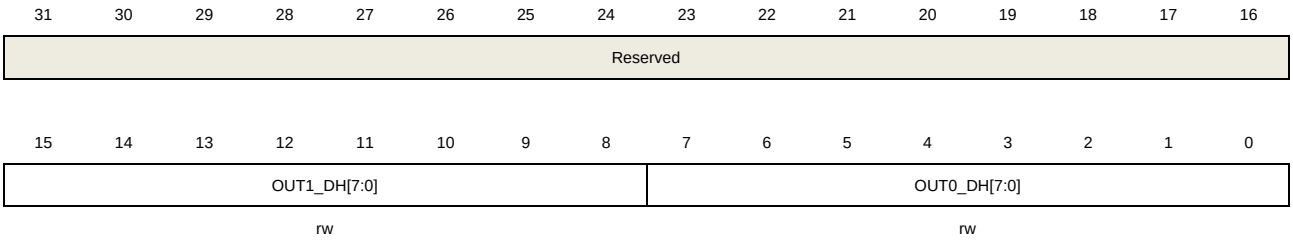


Bits	Fields	Descriptions
31:20	OUT1_DH[11:0]	DACx_OUT1 12-bit left-aligned data These bits specify the data that is to be converted by DACx_OUT1.
19:16	Reserved	Must be kept at reset value.
15:4	OUT0_DH[11:0]	DACx_OUT0 12-bit left-aligned data These bits specify the data that is to be converted by DACx_OUT0.
3:0	Reserved	Must be kept at reset value.

14.4.11. DACx 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).



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:8	OUT1_DH[7:0]	DACx_OUT1 8-bit right-aligned data These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT1.

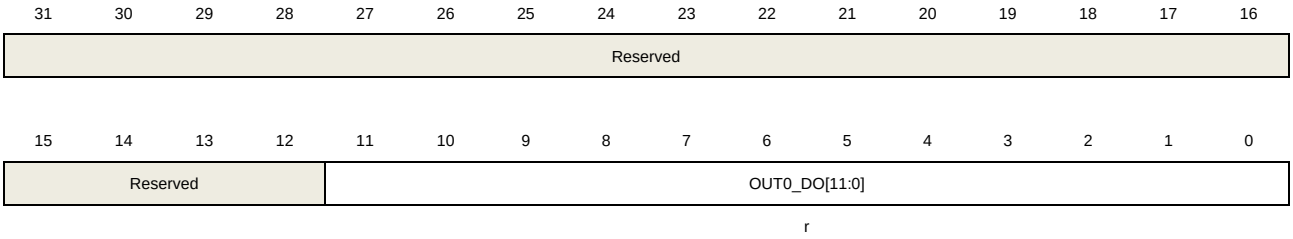
7:0	OUT0_DH[7:0]	DACx_OUT0 8-bit right-aligned data These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT0.

14.4.12. DACx_OUT0 data output register (DAC_OUT0_DO)

Address offset: 0x2C

Reset value: 0x0000 0000

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



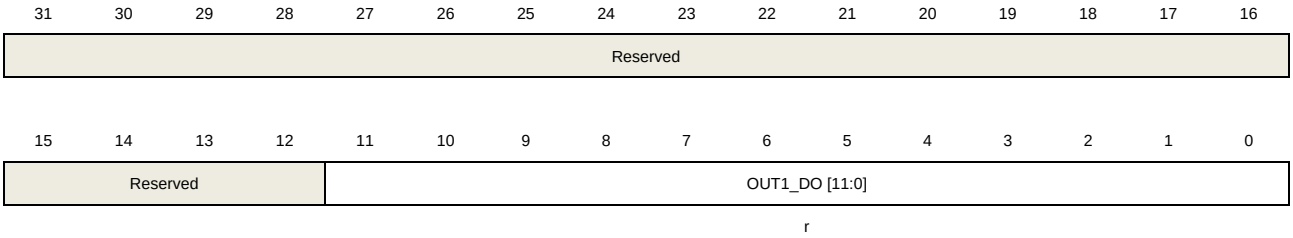
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	OUT0_DO[11:0]	DACx_OUT0 12-bit output data These bits, which are read only, storage the data that is being converted by DACx_OUT0.

14.4.13. DACx_OUT1 data output register (DAC_OUT1_DO)

Address offset: 0x30

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	OUT1_DO[11:0]	DACx_OUT1 12-bit output data These bits, which are read only, storage the data that is being converted by DACx_OUT1.

14.4.14. DACx status register 0 (DAC_STAT0)

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		DDUDR1	Reserved												
		rc_w1													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DDUDR0	Reserved												
		rc_w1													

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	DDUDR1	DACx_OUT1 DMA underrun flag This bit is set by hardware and cleared by software (by writing it to 1). 0: No underrun occurred. 1: Underrun occurred (Speed of DAC trigger is higher than the DMA transfer).
28:14	Reserved	Must be kept at reset value.
13	DDUDR0	DACx_OUT0 DMA underrun flag This bit is set by hardware and cleared by software (by writing it to 1). 0: No underrun occurred. 1: Underrun occurred (Speed of DAC trigger is higher than the DMA transfer).
12:0	Reserved	Must be kept at reset value.

14.4.15. DACx control register 1 (DAC_CTL1)

Address offset: 0x80

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													FIFO UDRIE1	FIFO OVRIE1	FIFO EN1
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													FIFO UDRIE0	FIFO OVRIE0	FIFO EN0
													rw	rw	rw

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value.
18	FIFOUDRIE1	DACx_OUT1 FIFO underflow interrupt enable 0: DACx_OUT1 FIFO underflow interrupt disabled. 1: DACx_OUT1 FIFO underflow interrupt enabled.
17	FIFOOVRIE1	DACx_OUT1 FIFO overflow interrupt enable 0: DACx_OUT1 FIFO overflow interrupt disabled. 1: DACx_OUT1 FIFO overflow interrupt enabled.
16	FIFOEN1	DACx_OUT1 data FIFO enable The DTEN1 bit should be set and DDMAEN1 should be reset, when FIFOEN1 is set. 0: DACx_OUT1 data FIFO disable. 1: DACx_OUT1 data FIFO enable.
15:3	Reserved	Must be kept at reset value.
2	FIFOUDRIE0	DACx_OUT0 FIFO underflow interrupt enable 0: DACx_OUT0 FIFO underflow interrupt disabled. 1: DACx_OUT0 FIFO underflow interrupt enabled.
1	FIFOOVRIE0	DACx_OUT0 FIFO overflow interrupt enable 0: DACx_OUT0 FIFO overflow interrupt disabled. 1: DACx_OUT0 FIFO overflow interrupt enabled
0	FIFOEN0	DACx_OUT0 data FIFO enable The DTEN0 bit should be set and DDMAEN0 should be reset, when FIFOEN0 is set. 0: DACx_OUT0 data FIFO disable. 1: DACx_OUT0 data FIFO enable.

14.4.16. DACx status register 1 (DAC_STAT1)

Address offset: 0x84

Reset value: 0x0002 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									FIFONUM1			FIFOUDR1	FIFO OVR1	FIFOE1	FIFOF1
										r		rc_w1	rc_w1	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									FIFONUM0			FIFO UDR0	FIFO OVR0	FIFOE0	FIFOF0
										r		rc_w1	rc_w1	r	r

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value.
22:20	FIFONUM1	DACx_OUT1 FIFO length 000: The length of data is 0 001: The length of data is 1 010: The length of data is 2 011: The length of data is 3 100: The length of data is 4 101~111: Reserved
19	FIFOUDR1	DACx_OUT1 FIFO underflow flag 0: DACx_OUT1 FIFO is not underflow. 1: DACx_OUT1 FIFO is underflow.
18	FIFOOVR1	DACx_OUT1 FIFO overflow flag 0: DACx_OUT1 FIFO is not overflow. 1: DACx_OUT1 FIFO is overflow.
17	FIFOE1	DACx_OUT1 FIFO empty flag 0: DACx_OUT1 FIFO is not empty. 1: DACx_OUT1 FIFO is empty.
16	FIFOF1	DACx_OUT1 FIFO full flag 0: DACx_OUT1 FIFO is not full. 1: DACx_OUT1 FIFO is full.
15:7	Reserved	Must be kept at reset value.
6:4	FIFONUM0	DACx_OUT0 FIFO length 000: The length of data is 0 001: The length of data is 1 010: The length of data is 2 011: The length of data is 3 100: The length of data is 4 101~111: Reserved
3	FIFOUDR0	DACx_OUT0 FIFO underflow flag 0: DACx_OUT0 FIFO is not underflow. 1: DACx_OUT0 FIFO is underflow.
2	FIFOOVR0	DACx_OUT0 FIFO overflow flag 0: DACx_OUT0 FIFO is not overflow. 1: DACx_OUT0 FIFO is overflow.
1	FIFOE0	DACx_OUT0 FIFO empty flag 0: DACx_OUT0 FIFO is not empty.

		1: DACx_OUT0 FIFO is empty.
0	FIFOF0	DACx_OUT0 FIFO full flag 0: DACx_OUT0 FIFO is not full. 1: DACx_OUT0 FIFO is full.

15. Comparator (CMP)

The CMP is only available on CL series.

15.1. Overview

The general purpose comparator CMP, can work either standalone (all terminal are available on I/Os) or together with the timers.

It can be used to provide a trigger source when an analog signal is in a certain condition.

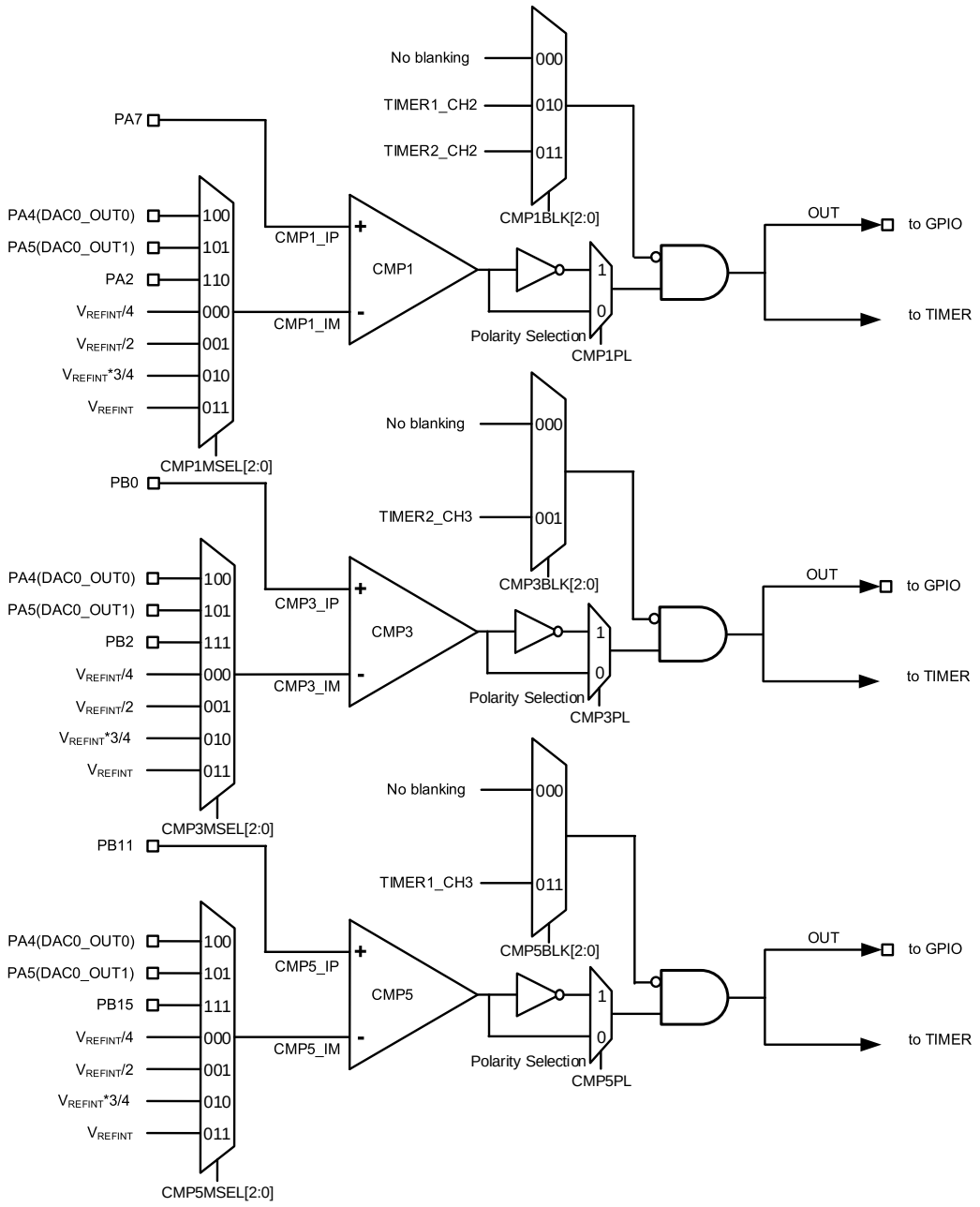
15.2. Characteristics

- Rail-to-rail comparators.
- Configurable analog input source.
  - DAC output.
  - Multiplexed I / O pins.
  - The whole or sub-multiple values of internal reference voltage.
- Outputs with blanking source.
- Outputs to I / O.
- Outputs to timers for triggering.

15.3. Function overview

The block diagram of CMP is shown below.

Figure 15-1. CMP block diagram



Note: V_{REFINT} is 1.2V.

15.3.1. CMP clock

The clock of the CMP which is connected to APB bus, is synchronous with PCLK.

15.3.2. CMP I / O configuration

These I / Os must be configured in analog mode in the GPIOs registers before they are selected as CMP inputs.

The CMP output can be redirected internally and externally simultaneously.

Refer to pin definitions in datasheet, and the CMP output can be connected to the corresponding I/O port via the alternate function of the GPIO.

CMP output internally connect to the TIMER and the connections between them are as follows:

- CMP output to the TIMER input channel.
- CMP output to the TIMER break.

Table 15-1. CMP inputs and outputs summary details the inputs and outputs of the CMP.

Table 15-1. CMP inputs and outputs summary

	CMP1	CMP3	CMP5
CMP non inverting inputs connected to I / Os	PA7	PB0	PB11
CMP inverting inputs connected to I / Os	PA2 PA4 PA5	PA4 PA5 PB2	PA4 PA5 PB15
CMP inverting inputs connected to internal signals	V_{REFINT} / 4 V_{REFINT} / 2 V_{REFINT} * 3 / 4 V_{REFINT} DAC0_OUT0 DAC0_OUT1	V_{REFINT} / 4 V_{REFINT} / 2 V_{REFINT} * 3 / 4 V_{REFINT} DAC0_OUT0 DAC0_OUT1	V_{REFINT} / 4 V_{REFINT} / 2 V_{REFINT} * 3 / 4 V_{REFINT} DAC0_OUT0 DAC0_OUT1
CMP outputs connected to I / Os	PA2 PA12 PB9 PE8 PE13	PB1 PE9 PE12	PA10 PC6 PE10 PE11
CMP outputs connected to internal signals	TIMER0_CH0 TIMER1_CH3 TIMER2_CH0 SHRTIMER_EXEV0 SHRTIMER_EXEV5	TIMER2_CH2 SHRTIMER_EXEV1 SHRTIMER_EXEV6	TIMER1_CH1 SHRTIMER_EXEV2 SHRTIMER_EXEV7
CMP outputs(motor control protection)	TIMER0 BRKIN		

Note: The output of CMP1/3/5 is directly connected to the SHRTIMER peripheral.

15.3.3. CMP register write protection

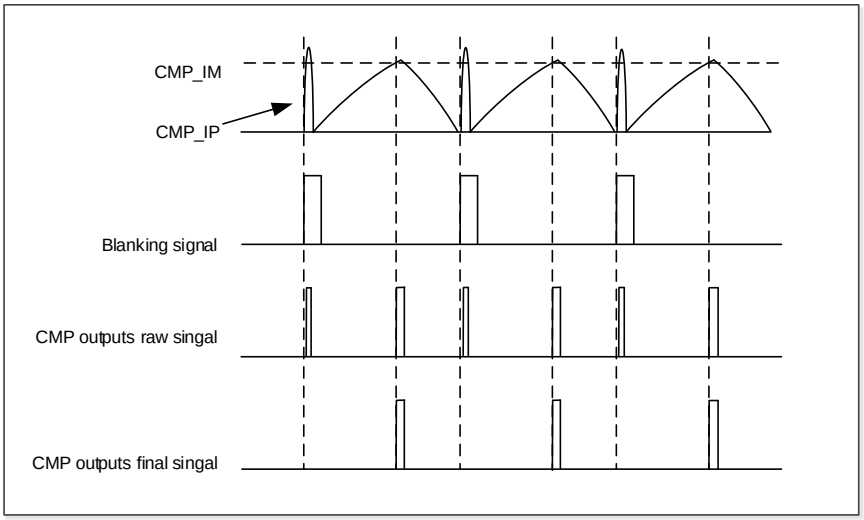
The CMP control and status register (CMPx_CS) can be protected from writing by setting CMPxLK bit to 1. The CMPx_CS register, including the CMPxLK bit will be read-only, and can only be reset by the MCU reset.

15.3.4. CMP output blanking

CMP output blanking function can be used to avoid interference of short pulses in the input signal to CMP output signal. If the CMPxBLK[2:0] bits in the CMPx_CS register are setting to an available value, the CMP output final signal is obtained by ANDing the complementary signal of the selected blanking signal with the raw output of the comparator. The blanking function can be used for false overcurrent detection in motor control applications.

Figure 15-2. The CMP outputs signal blanking shows the comparator output blank function.

Figure 15-2. The CMP outputs signal blanking



15.4. Register definition

CMP base address：0x4001 7C00

15.4.1. CMP1 Control / status register (CMP1_CS)

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
CMP1LK	CMP1O	Reserved							CMP1MSEL[3]	Reserved	CMP1BLK[2:0]			Reserved	
rwo	r								rw		rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1PL	Reserved	CMP1OSEL[3:0]				Reserved			CMP1MSEL[2:0]			Reserved			CMP1EN
rw		rw							rw						rw

Bits	Fields	Descriptions
31	CMP1LK	CMP1 lock This bit allows to have all control bits of CMP1 as read-only. It can only be set once by software and cleared by a system reset. 0: CMP1_CS bits are read-write 1: CMP1_CS bits are read-only
30	CMP1O	CMP1 output state This bit is a copy of CMP1 output state, which is read only. 0: Non-inverting input below inverting input and the output is low 1: Non-inverting input above inverting input and the output is high
29:23	Reserved	Must be kept at reset value.
22	CMP1MSEL[3]	Bit 3 of CMP1MSEL see bits 6:4 of CMP1MSEL
21	Reserved	Must be kept at reset value.
20:18	CMP1BLK[2:0]	CMP1 output blanking source This bit is used to select which Timer output controls the CMP1 output blanking. 000: No blanking 001: Reserved 010: Select TIMER1_CH2 output compare signal as blanking source 011: Select TIMER2_CH2 output compare signal as blanking source 100~111: Reserved
17:16	Reserved	Must be kept at reset value.

15	CMP1PL	Polarity of CMP1 output This bit is used to select the polarity of CMP1 output. 0: Output is not inverted 1: Output is inverted
14	Reserved	Must be kept at reset value
13:10	CMP1OSEL[3:0]	CMP1 output selection These bits are used to select the destination of the CMP1 output. 0000: no selection 0001: TIMER0 break input 0010~0110: Reserved 0111: TIMER0 CH0 input capture 1000: TIMER1 CH3 input capture 1001: Reserved 1010: TIMER2 CH0 input capture 1011~1111: Reserved **Note:** It is recommended to enable CMP first, and then configure the timer channel, when using TIMER to capture the output signal of the comparator.
9:7	Reserved	Must be kept at reset value
6:4	CMP1MSEL[2:0]	CMP1_IM input selection These bits, together with bit 22, are used to select the source connected to the CMP1_IM input of the CMP1. 0000: V_{REFINT} / 4 0001: V_{REFINT} / 2 0010: V_{REFINT} * 3 / 4 0011: V_{REFINT} 0100: PA4 (DAC0_OUT0) 0101: PA5 (DAC0_OUT1) 0110: PA2 0111~1111: Reserved
3:1	Reserved	Must be kept at reset value
0	CMP1EN	CMP1 enable 0: CMP1 disabled 1: CMP1 enabled

15.4.2. CMP3 Control / status register (CMP3_CS)

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
CMP3LK	CMP3O	Reserved							CMP3MSEL[3]	Reserved	CMP3BLK[2:0]			Reserved	
rwo	r								rw		rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3PL	Reserved	CMP3OSEL[3:0]				Reserved			CMP3MSEL[2:0]			Reserved			CMP3EN
rw		rw							rw						rw

Bits	Fields	Descriptions
31	CMP3LK	CMP3 lock This bit allows to have all control bits of CMP3 as read-only. It can only be set once by software and cleared by a system reset. 0: CMP3_CS bits are read-write 1: CMP3_CS bits are read-only
30	CMP3O	CMP3 output state This bit is a copy of CMP3 output state, which is read only. 0: Non-inverting input below inverting input and the output is low 1: Non-inverting input above inverting input and the output is high
29:23	Reserved	Must be kept at reset value.
22	CMP3MSEL[3]	Bit 3 of CMP3MSEL see bits 6:4 of CMP3MSEL
21	Reserved	Must be kept at reset value.
20:18	CMP3BLK[2:0]	CMP3 output blanking source This bit is used to select which Timer output controls the CMP3 output blanking. 000: No blanking 001: Select TIMER2_CH3 output compare signal as blanking source 010~111: Reserved
17:16	Reserved	Must be kept at reset value.
15	CMP3PL	Polarity of CMP3 output This bit is used to select the polarity of CMP3 output. 0 : Output is not inverted 1 : Output is inverted
14	Reserved	Must be kept at reset value
13:10	CMP3OSEL[3:0]	CMP3 output selection These bits are used to select the destination of the CMP3 output. 0000: no selection 0001: TIMER0 break input 0010~0101: Reserved 0110: TIMER2 CH2 input capture

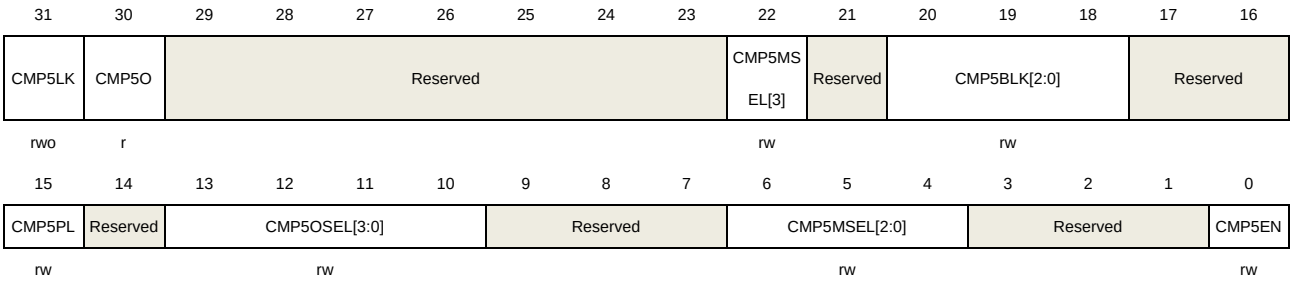
		0111~1111: Reserved **Note:** It is recommended to enable CMP first, and then configure the timer channel, when using TIMER to capture the output signal of the comparator.
9:7	Reserved	Must be kept at reset value
6:4	CMP3MSEL[2:0]	CMP3_IM input selection These bits, together with bit 22, are used to select the source connected to the CMP3_IM input of the CMP3. 0000: V_{REFINT} / 4 0001: V_{REFINT} / 2 0010: V_{REFINT} * 3 / 4 0011: V_{REFINT} 0100: PA4 (DAC0_OUT0) 0101: PA5 (DAC0_OUT1) 0110: Reserved 0111: PB2 1000~1111: Reserved
3:1	Reserved	Must be kept at reset value
0	CMP3EN	CMP3 enable 0: CMP3 disabled 1: CMP3 enabled

15.4.3. CMP5 Control / status register (CMP5_CS)

Address offset: 0x30

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31	CMP5LK	CMP5 lock This bit allows to have all control bits of CMP5 as read-only. It can only be set once by software and cleared by a system reset. 0: CMP5_CS bits are read-write 1: CMP5_CS bits are read-only

30	CMP5O	CMP5 output state This bit is a copy of CMP5 output state, which is read only. 0: Non-inverting input below inverting input and the output is low 1: Non-inverting input above inverting input and the output is high
29:23	Reserved	Must be kept at reset value.
22	CMP5MSEL[3]	Bit 3 of CMP5MSEL see bits 6:4 of CMP5MSEL
21	Reserved	Must be kept at reset value.
20:18	CMP5BLK[2:0]	CMP5 output blanking source This bit is used to select which Timer output controls the CMP5 output blanking. 000: No blanking 001: Reserved 010: Reserved 011: Select TIMER1_CH3 output compare signal as blanking source 100~111: Reserved
17:16	Reserved	Must be kept at reset value.
15	CMP5PL	Polarity of CMP5 output This bit is used to select the polarity of CMP5 output. 0 : Output is not inverted 1 : Output is inverted
14	Reserved	Must be kept at reset value
13:10	CMP5OSEL[3:0]	CMP5 output selection These bits are used to select the destination of the CMP5 output. 0000: no selection 0001: TIMER0 break input 0010~0101:Reserved 0110: TIMER1 channel1 input capture 0111~1111: Reserved **Note:** It is recommended to enable CMP first, and then configure the timer channel, when using TIMER to capture the output signal of the comparator.
9:7	Reserved	Must be kept at reset value
6:4	CMP5MSEL[2:0]	CMP5_IM input selection These bits, together with bit 22, are used to select the source connected to the CMP5_IM input of the CMP5. 0000: V_{REFINT} / 4 0001: V_{REFINT} / 2 0010: V_{REFINT} * 3 / 4 0011: V_{REFINT} 0100: PA4 (DAC0_OUT0)

		0101: PA5 (DAC0_OUT1) 0110: Reserved 0111: PB15 1000~1111: Reserved
3:1	Reserved	Must be kept at reset value
0	CMP5EN	CMP5 enable 0: CMP5 disabled 1: CMP5 enabled

16. 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 when the internal counter reaches a given value. The watchdog timer counter can be stopped while the processor is in the debug mode.

16.1. Free watchdog timer (FWDGT)

16.1.1. 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.

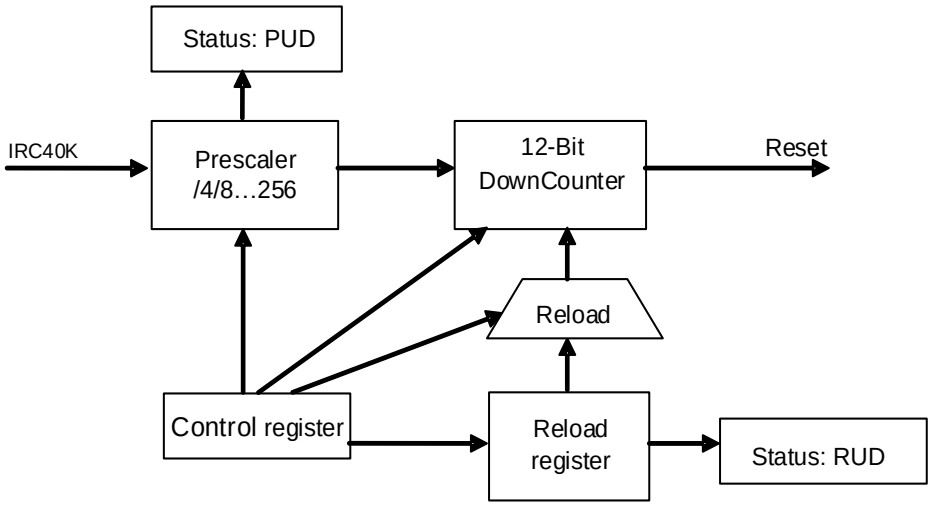
16.1.2. Characteristics

- 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 or not when power on.
- FWDGT debug mode, the FWDGT can stop or continue to work in debug mode.

16.1.3. Function overview

The free watchdog consists of an 8-stage prescaler and a 12-bit down-counter. ***Figure 16-1. Free watchdog block diagram*** shows the functional block of the free watchdog module.

Figure 16-1. Free watchdog block diagram



The free watchdog is enabled by writing the value (0xCCCC) to the control register (FWDGT_CTL), then the counter starts counting down. When the counter reaches the value (0x000), there will be a reset.

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 when power on if the hardware free watchdog bit in the device option bits is set. To avoid a reset, the software should reload the counter before the counter reaches (0x000).

The FWDGT_PSC register and the FWDGT_RLD register are written 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 ongoing, 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®-M33 core halted (Debug mode). The FWDGT stops in Debug mode if the FWDGT_HOLD bit is set.

Table 16-1. 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	0.025	409.525
1 / 8	001	0.025	819.025
1 / 16	010	0.025	1638.025
1 / 32	011	0.025	3276.025
1 / 64	100	0.025	6552.025

Prescaler divider	PSC[2:0] bits	Min timeout (ms) RLD[11:0]=0x000	Max timeout (ms) RLD[11:0]=0xFFF
1 / 128	101	0.025	13104.025
1 / 256	110 or 111	0.025	26208.025

The FWDGT timeout can be more accurate by calibrating the IRC40K.

Note: When after the execution of watchdog reload operation, if the MCU needs enter the deepsleep / standby mode immediately, more than 3 IRC40K clock intervals must be inserted in the middle of reload and deepsleep / standby mode commands by software setting.

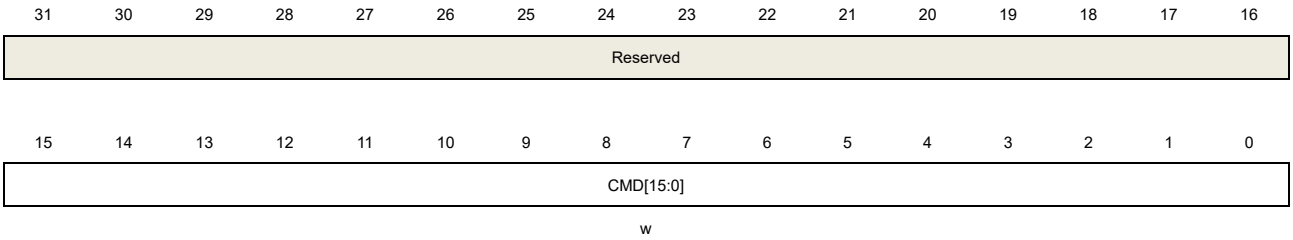
16.1.4. Register definition

FWDGT base address: 0x4000 3000

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.

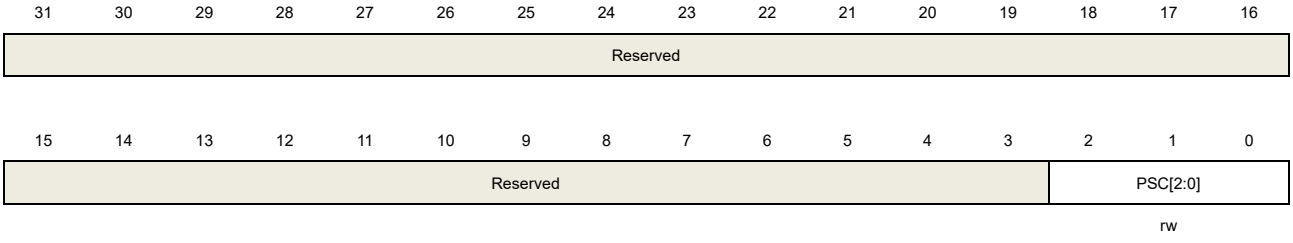


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CMD[15:0]	Write only. Several different fuctions are realized by writing these bits with different values: 0x5555: Disable the FWDGT_PSC and FWDGT_RLD write protection. 0xCCCC: Start the free watchdog counter. When the counter 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.



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

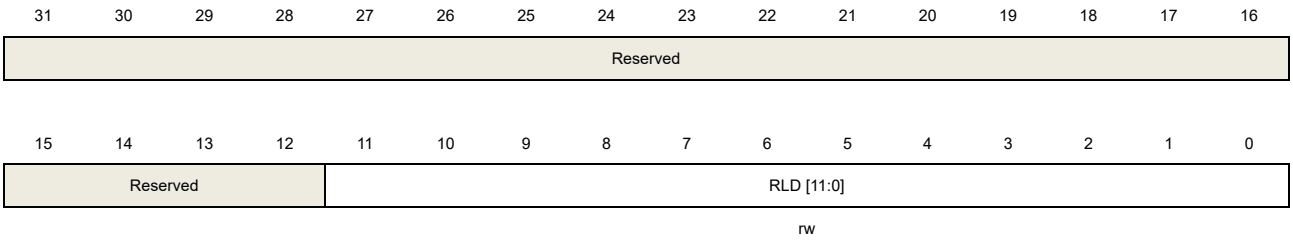
If several prescaler values are used by the application, it is mandatory to wait until PUD bit is reset before changing the prescaler value. However, after updating the prescaler value it is not necessary to wait until PUD is reset before continuing code execution (Before entering low-power mode, it is necessary to wait until PUD is reset).

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



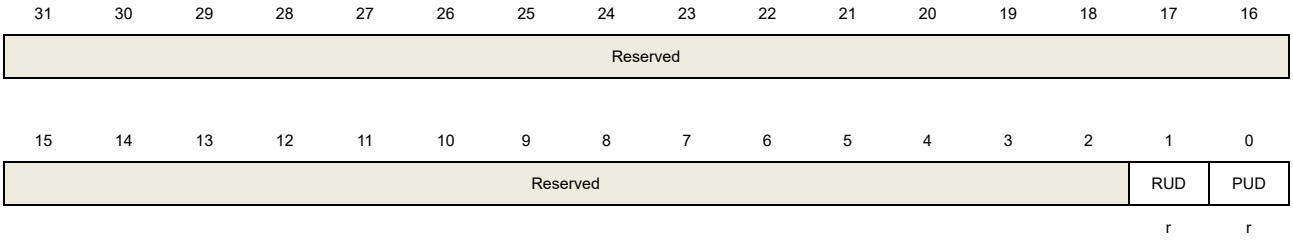
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	RLD[11:0]	Free watchdog timer counter reload value. Write 0xAAAA in the FWDGT_CTL register will reload the FWDGT counter with the RLD value. These bits are write-protected. Write 0x5555 in the FWDGT_CTL register before writing 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 (Before entering low-power mode, it is necessary to wait until RUD is reset).

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.



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.

16.2. Window watchdog timer (WWDGT)

16.2.1. 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 down counter reduces progressively. The watchdog timer causes a reset when the counter reached 0x3F (the CNT[6] bit has been cleared). The watchdog timer also causes a reset when 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. Interrup occurs if it is enable.

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.

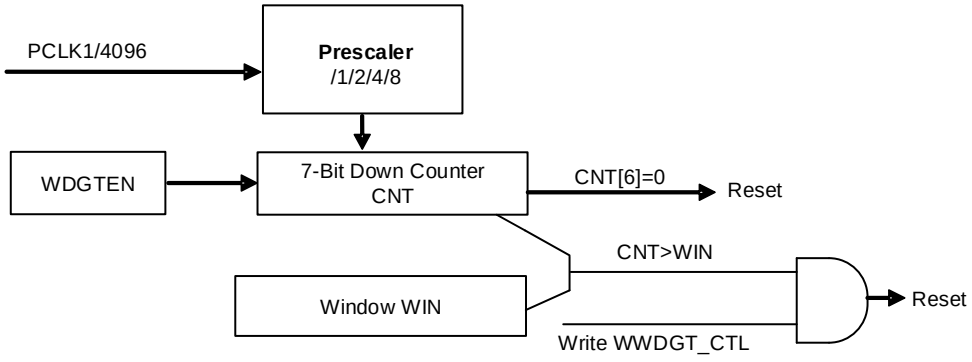
16.2.2. Characteristics

- 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): the watchdog is started and the interrupt is enabled, the interrupt occurs when the counter reaches 0x40.
- WWDGT debug mode, the WWDGT can stop or continue to work in debug mode.

16.2.3. Function overview

If the window watchdog timer is enabled (set the WDGTEN bit in the WWDGT_CTL), the watchdog timer causes a reset when the counter reaches 0x3F (the CNT[6] bit has been cleared), or the counter is refreshed before the counter reaches the window register value.

Figure 16-2. Window watchdog timer block diagram



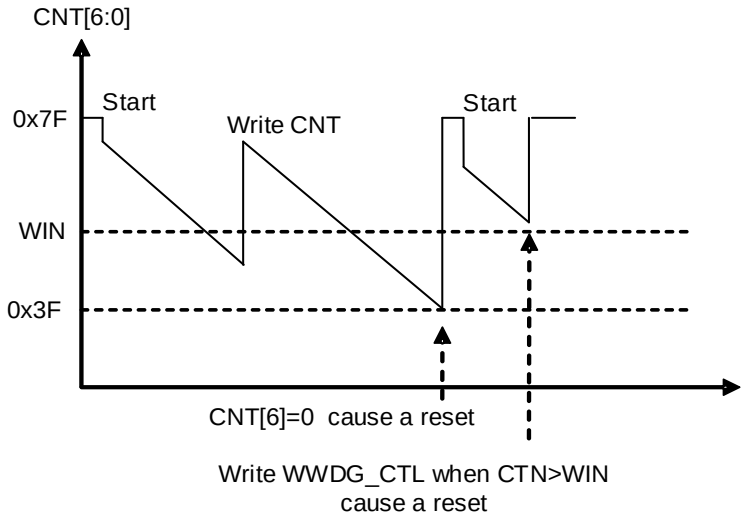
The window watchdog timer is always disabled after power on reset. The software starts the watchdog by setting the WDGTEN bit in the WWDGT_CTL register. When window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F (it implies that the CNT[6] bit should be set). The CNT[5:0] determine the maximum time interval between two reloading. The count down 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 down counter. The 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 will be generated when the counter reaches 0x40. 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 16-3. Window watchdog timing diagram



Calculate the WWDGT timeout by using the formula below.

$$t_{WWDGT}=t_{PCLK1} \times 4096 \times 2^{PSC} \times (CNT[5:0]+1) \quad (ms) \qquad (16\text{-}1)$$

where:

t_{WWDGT}: WWDGT timeout

t_{PCLK1}: APB1 clock period measured in ms

The table below shows the minimum and maximum values of the t_{WWDGT}.

Table 16-2. Min/max timeout value at 100 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	40.96 µs	2.62 ms
1 / 2	01	81.92 µs	5.24 ms
1 / 4	10	163.84 µs	10.49 ms
1 / 8	11	327.68 µs	20.97 ms

If the WWDGT_HOLD bit in DBG module is cleared, the WWDGT continues to work even the Cortex®-M33 core halted (Debug mode). While the WWDGT_HOLD bit is set, the WWDGT stops in Debug mode.

16.2.4. Register definition

WWDGT base address: 0x4000 2C00

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. If the bit is set, an interrupt occurs when the counter

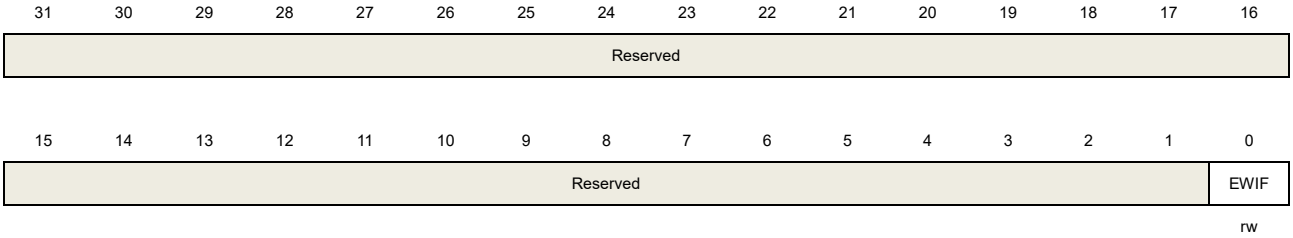
		reaches 0x40. It can be cleared by a hardware reset or software reset by setting the WWDGTRST bit of the RCU module. A write operation of '0' has no effect.
8:7	PSC[1:0]	Prescaler. The time base of the watchdog timer counter 00: (PCLK1 / 4096) / 1 01: (PCLK1 / 4096) / 2 10: (PCLK1 / 4096) / 4 11: (PCLK1 / 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).



Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	EWIF	Early wakeup interrupt flag. When the counter reaches 0x40, 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.

17. Real-time clock (RTC)

17.1. 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.

17.2. Characteristics

- 32-bit programmable counter for counting elapsed time
- Programmable prescaler: Max division factor is up to 2^{20}
- Separate clock domains:
  - PCLK1 clock domain
  - RTC clock domain (this clock must be at least 4 times slower than the PCLK1 clock)
- RTC clock source:
  - HXTAL clock divided by 128
  - LXTAL oscillator clock
  - IRC40K oscillator clock
- Maskable interrupt source:
  - Alarm interrupt
  - Second interrupt
  - Overflow interrupt

17.3. Function overview

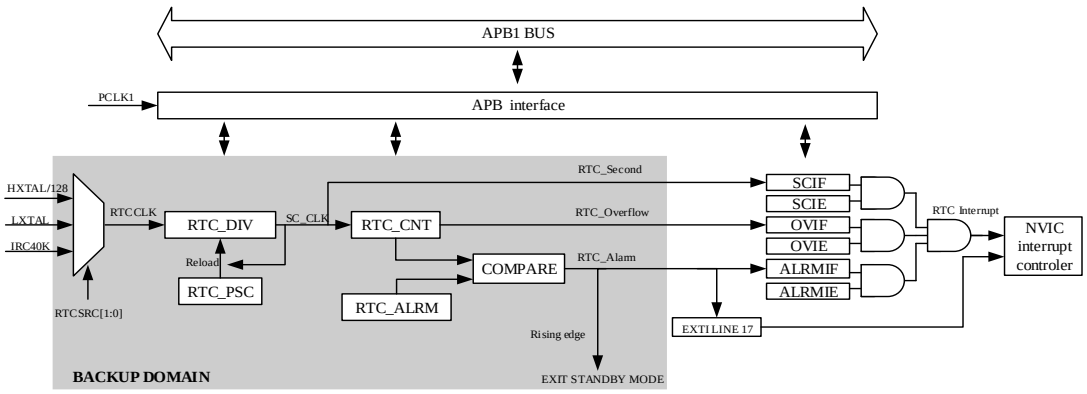
The RTC circuits consist of two major units: APB interface located in PCLK1 clock domain and RTC core located in RTC clock domain.

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 17-1. Block diagram of RTC



17.3.1. 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).

17.3.2. 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.

17.3.3. 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:

1. Wait until the value of LWOFF bit in the RTC_CTL register sets to '1';
2. Enter Configuration mode by setting the CMF bit in the RTC_CTL register;
3. Write to the RTC registers;
4. Exit Configuration mode by clearing the CMF bit in the RTC_CTL register;
5. Wait until the value of LWOFF bit in the RTC_CTL register sets to '1'.

17.3.4. 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 17-2. RTC second and alarm waveform example (RTC_PSC = 3, RTC_ALRM = 2)

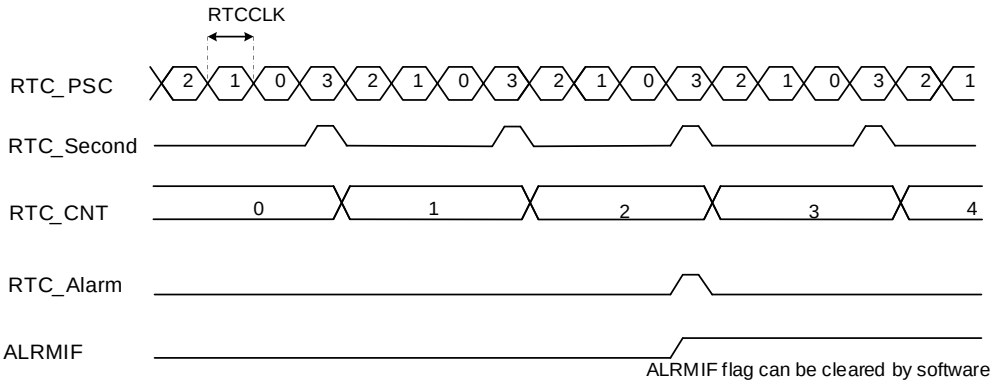
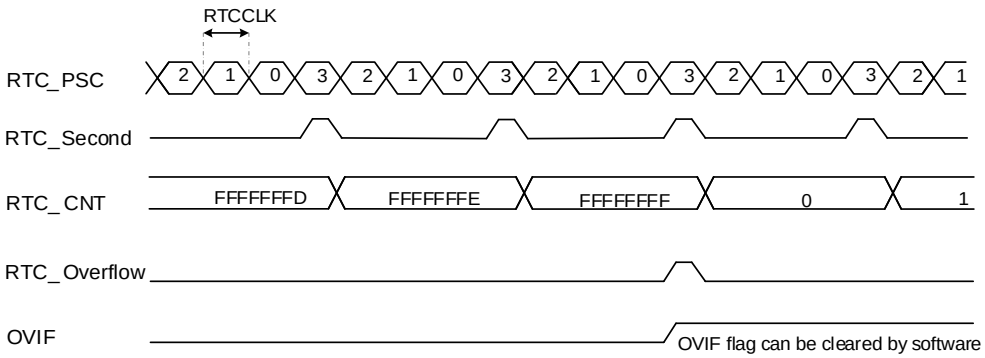


Figure 17-3. RTC second and overflow waveform example (RTC_PSC= 3)



17.4. RTC Register

RTC base address: 0x4000 2800

17.4.1. 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

17.4.2. 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															

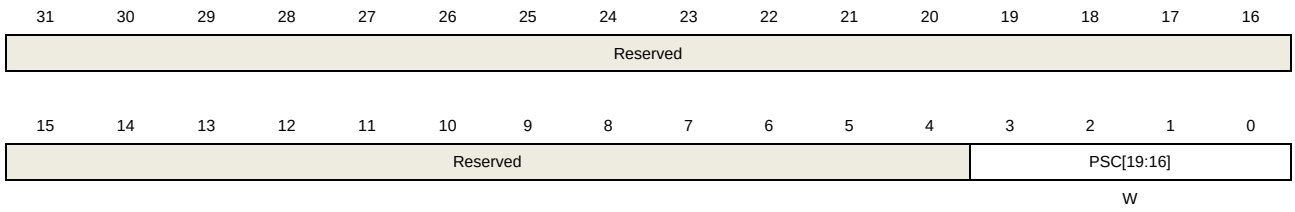
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										LWOFF	CMF	RSYNF	OVIF	ALRMIF	SCIF
										r	rw	rc_w0	rc_w0	rc_w0	rc_w0

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.

17.4.3. 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)



Bits	Fields	Descriptions

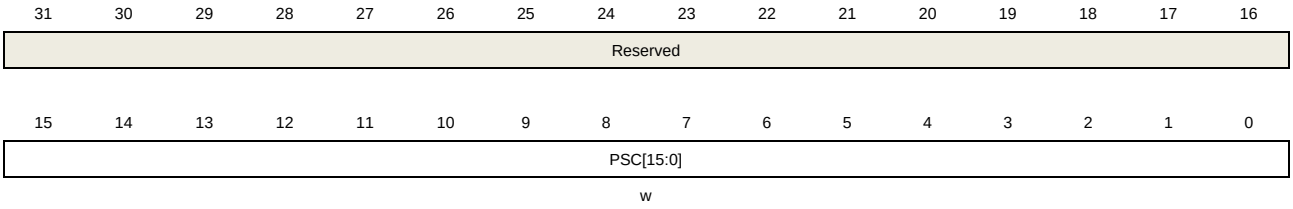
31:4	Reserved	Must be kept at reset value.
3:0	PSC[19:16]	RTC prescaler value high

17.4.4. 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)



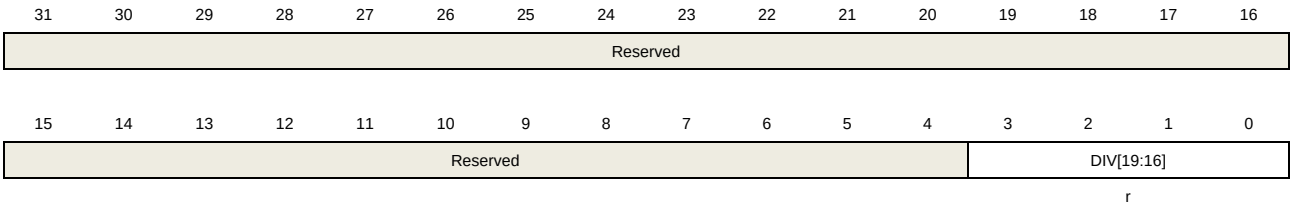
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	PSC[15:0]	RTC prescaler value low The frequency of SC_CLK is the RTCCLK frequency divided by (PSC[19:0]+1).

17.4.5. 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)



Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.
3:0	DIV[19:16]	RTC divider value high

17.4.6. 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)

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
DIV[15:0]															
r															

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.

17.4.7. 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)

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[31:16]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CNT[31:16]	RTC counter value high

17.4.8. 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)

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

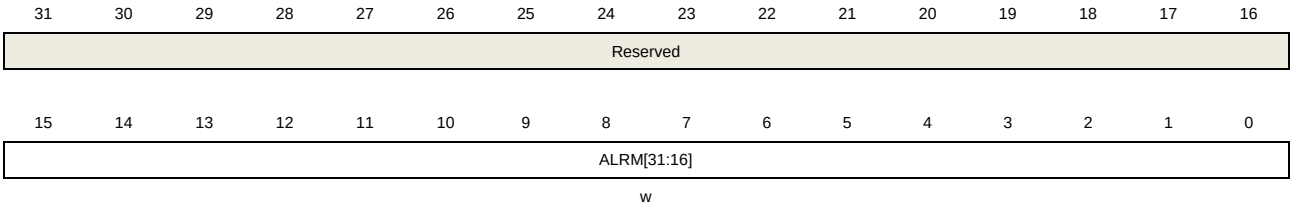
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CNT[15:0]	RTC counter value low

17.4.9. 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)

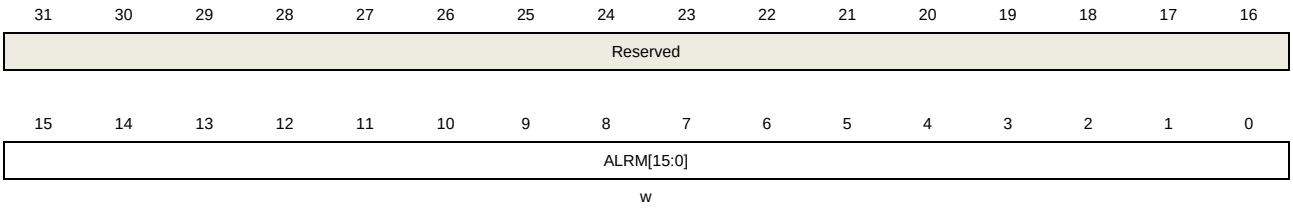


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	ALRM[31:16]	RTC alarm value high

17.4.10. 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)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	ALRM[15:0]	RTC alarm value low

18. Timer (TIMERx)

Table 18-1. Timers (TIMERx) are divided into five sorts

TIMER	TIMER0/7	TIMER1~4	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	32-bit(TIMER1) 16-bit(TIMER2~4)	16-bit	16-bit	16-bit
Count mode	UP,DOWN, Center-aligned	UP,DOWN, Center-aligned	UP ONLY	UP ONLY	UP ONLY
Repetition	●	×	×	×	×
CH Capture/ Compare	4	4	2	1	0
Complementary & Dead-time	●	×	×	×	×
Break	●	×	×	×	×
Single Pulse	●	●	●	×	●
Quadrature Decoder	●	●	×	×	×
Master-slave management	●	●	●	×	×
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;

18.1. Advanced timer (TIMERx, x=0, 7)

18.1.1. 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 is suitable 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.

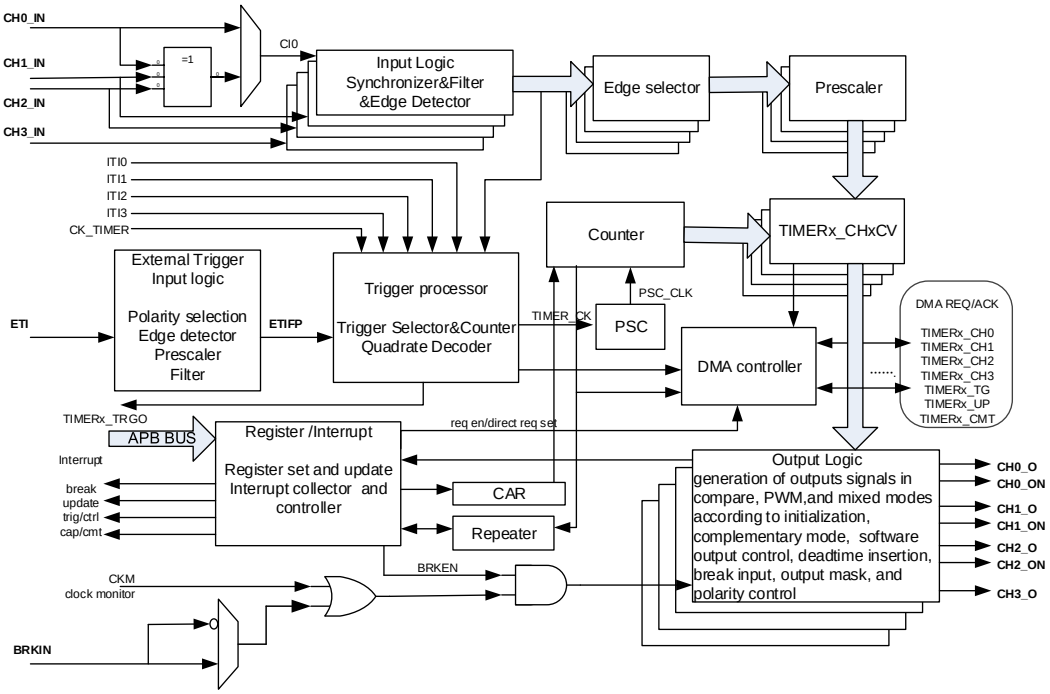
18.1.2. Characteristics

- 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 management.

18.1.3. Block diagram

Figure 18-1. Advanced timer block diagram provides details of the internal configuration of the advanced timer.

Figure 18-1. Advanced timer block diagram



18.1.4. Function overview

Clock source configuration

The advanced 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]).

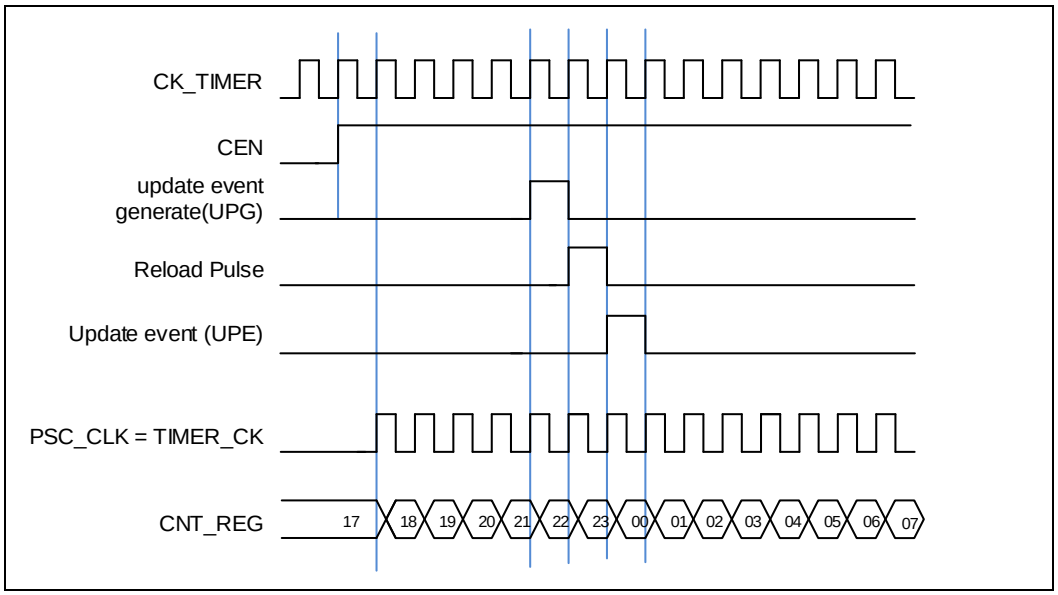
- SMC [2:0] == 3'b000. Internal clock CK_TIMER is selected as timer clock source which is from module RCU.

The default clock source is the CK_TIMER for driving the counter prescaler when the 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 SMC [2:0] in the TIMERx_SMCFG register are setting 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 SMC [2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK_TIMER is the counter prescaler driving clock source.

Figure 18-2. Timing chart of 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.

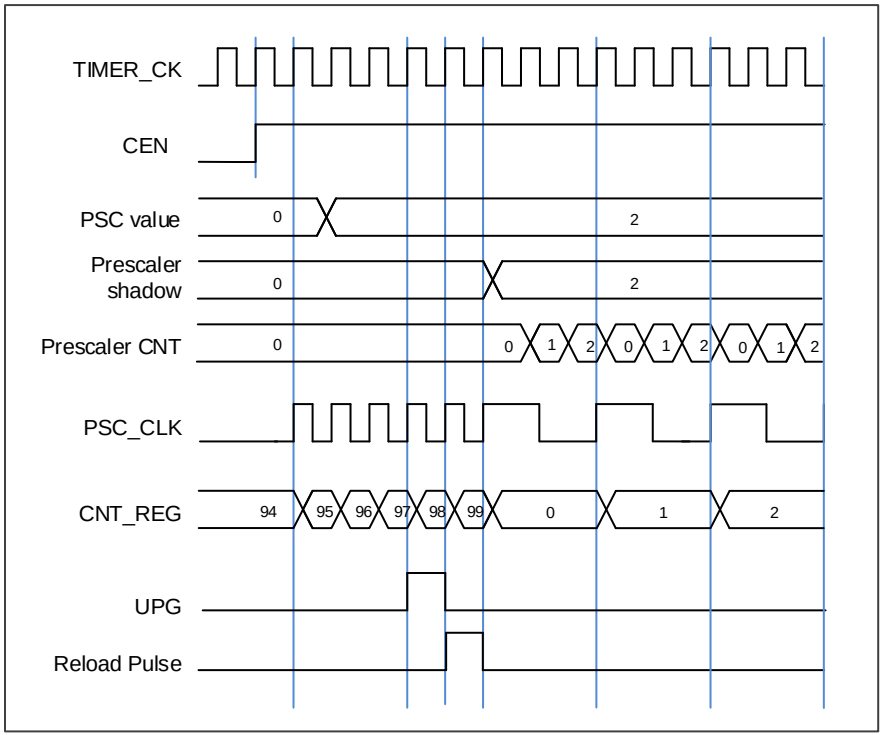
- SMC1== 1'b1 (external clock mode 1). External input is selected as timer clock source (ETI)

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.

Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 18-3. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again and an overflow event will be generated. In addition, the update events will be generated after (TIMERx_CREP+1) times of overflow events. 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 shadow registers (repetition counter, auto reload register, prescaler register) are updated.

Figure 18-4. Timing chart of up counting mode, PSC=0/2 and ***Figure 18-5. Timing chart of up counting mode, change TIMERx_CAR ongoing*** show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

Figure 18-4. Timing chart of up counting mode, PSC=0/2

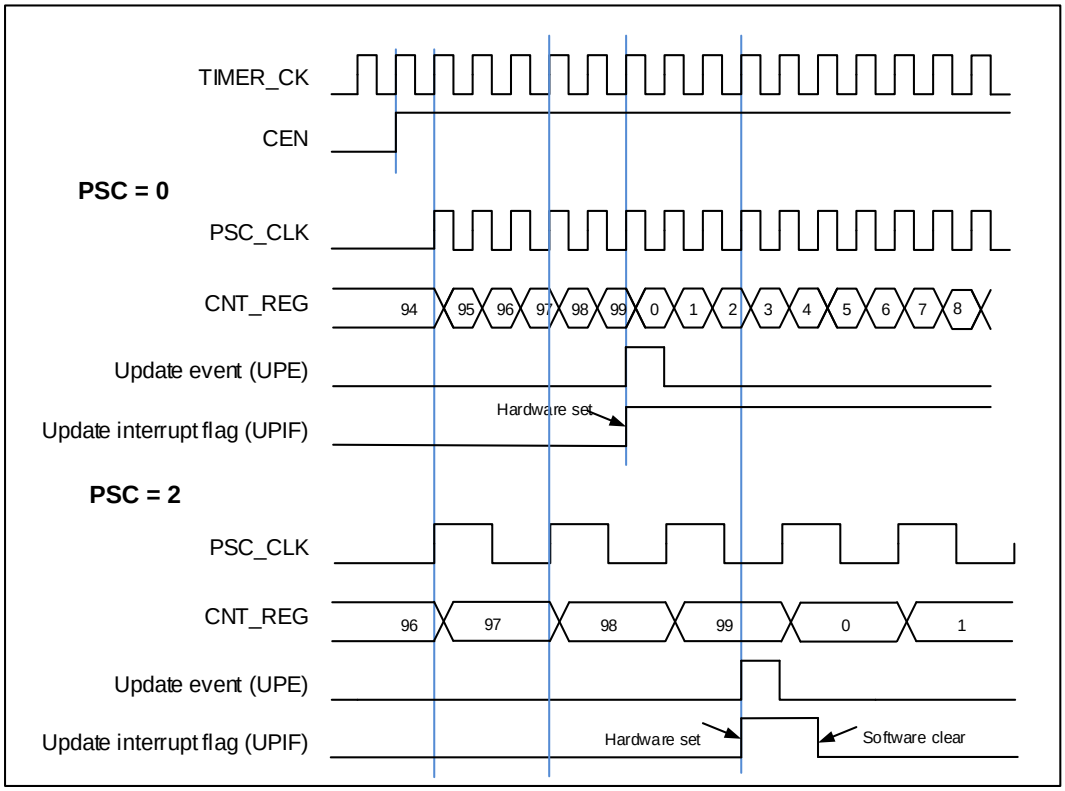
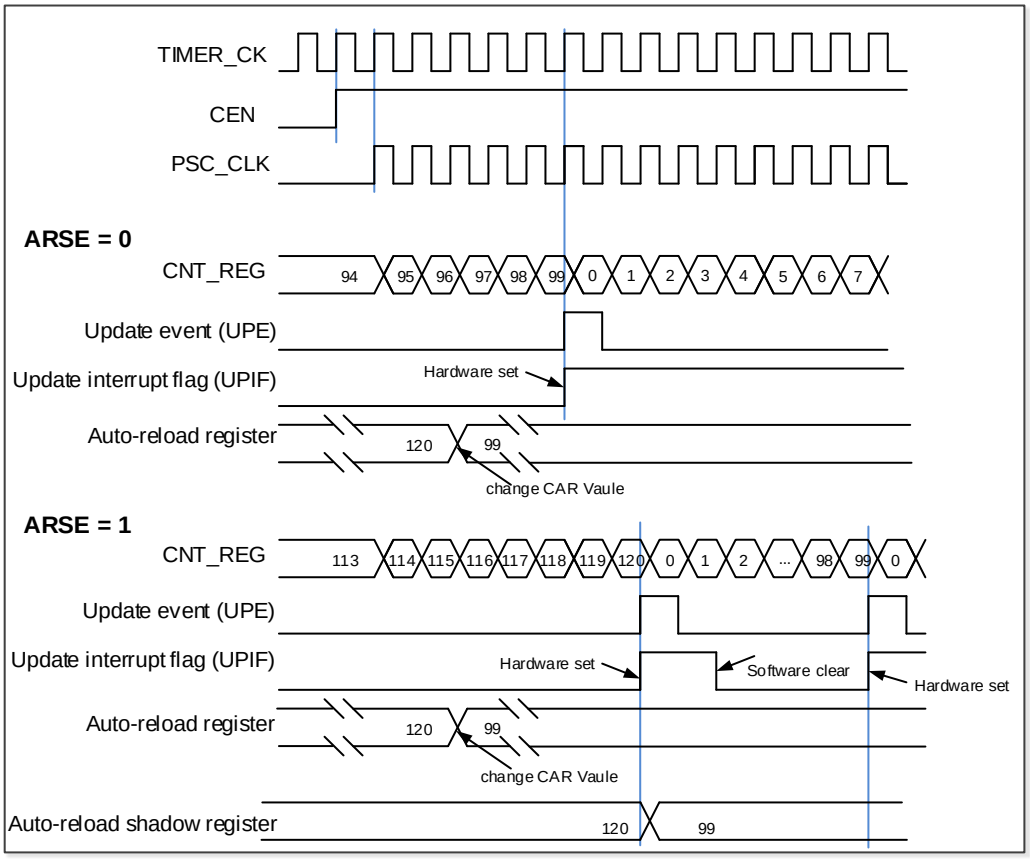


Figure 18-5. Timing chart of up counting mode, change TIMERx_CAR ongoing



Counter down counting

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 the counter will start counting down from the counter-reload value again and an underflow event will be generated. In addition, the update event will be generated after (TIMERx_CREP+1) times of 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 set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

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

Figure 18-6. Timing chart of down counting mode, PSC=0/2 and *Figure 18-7. Timing chart of down counting mode, change TIMERx_CAR ongoing* show some examples of the counter behavior in different clock frequencies when TIMERx_CAR=0x99.

Figure 18-6. Timing chart of down counting mode, PSC=0/2

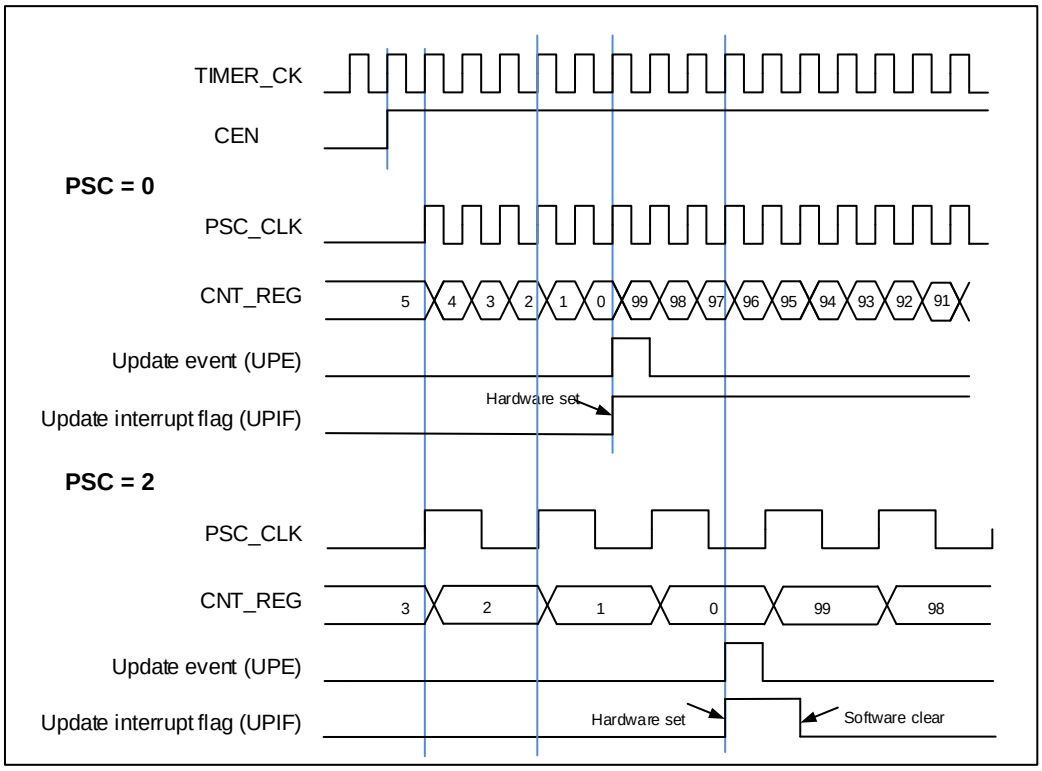
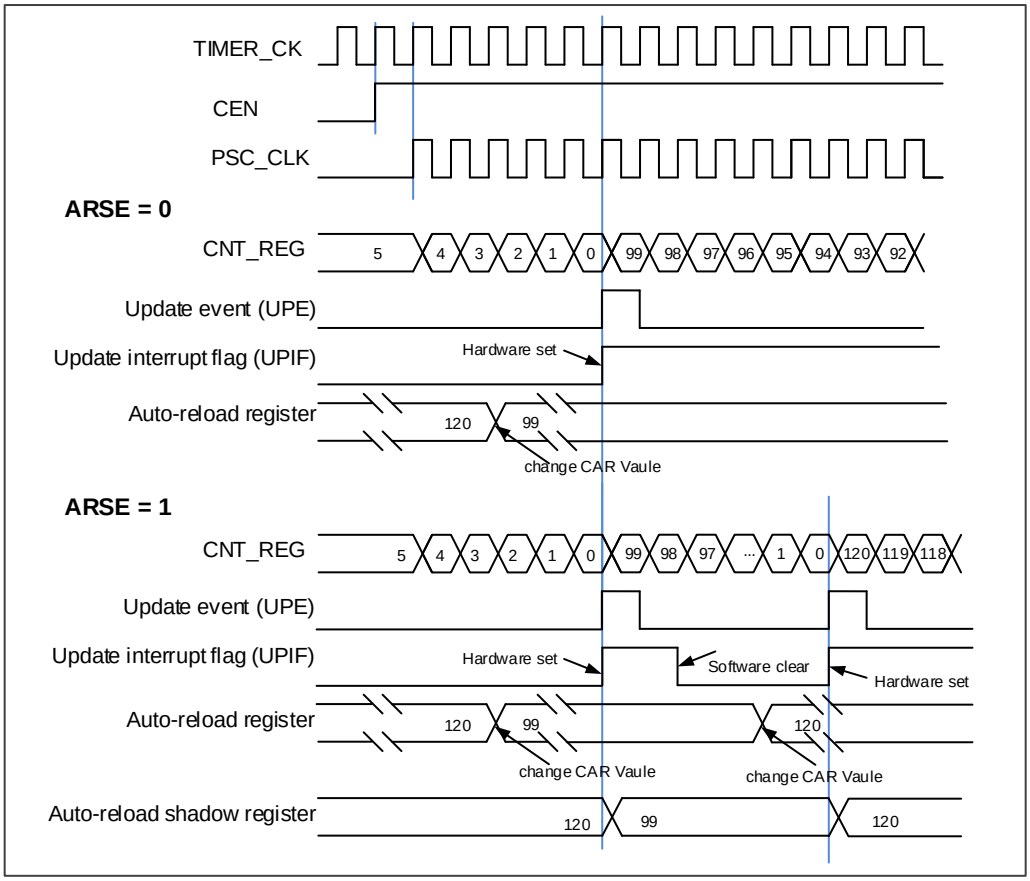


Figure 18-7. Timing chart of down counting mode, change TIMERx_CAR ongoing



Counter center-aligned counting

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.

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.

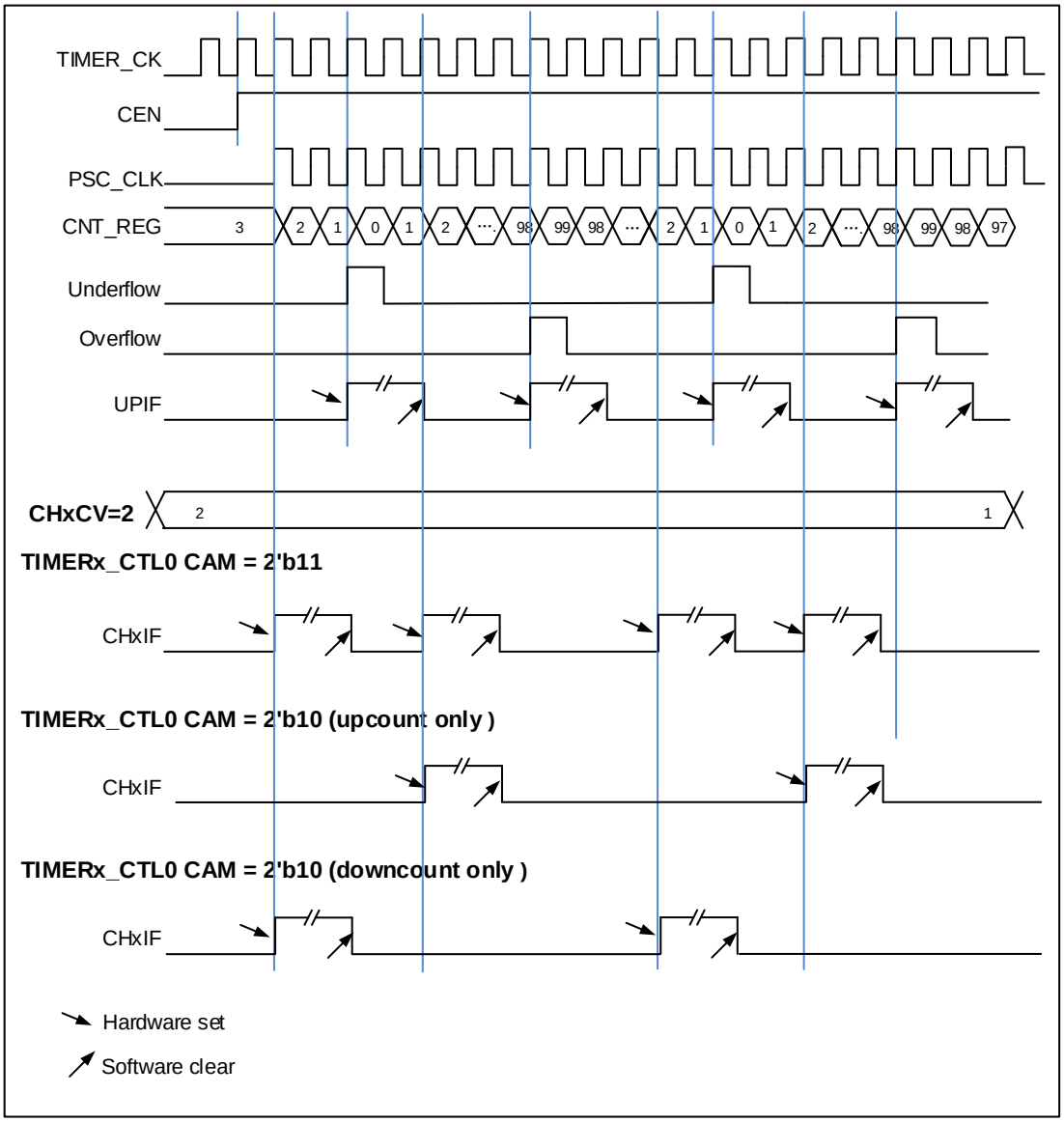
The UPIF bit in the TIMERx_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 18-8. Timing chart of center-aligned counting mode.***

If set the UPDIS bit in the TIMERx_CTL0 register, the update event is disabled.

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

Figure 18-8. Timing chart of center-aligned counting mode show some examples of the counter behavior when TIMERx_CAR=0x99. TIMERx_PSC=0x0

Figure 18-8. Timing chart of center-aligned counting mode



Update event (from overflow/underflow) rate configuration

The rate of update events generation (from overflow and underflow events) can be configured by the TIMERx_CREP register. 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 (does not exist in down counting mode) and underflow (does not exist in up counting 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.

The new written CREP value will not take effect until the next update event. When the value

of CREP is odd, and the counter is counting in center-aligned mode, the update event is generated (on overflow or underflow) depending on when the written CREP value takes effect. If an update event is generated by software after writing an odd number to CREP, the update events will be generated on the underflow. If the next update event occurs on overflow after writing an odd number to CREP, then the subsequent update events will be generated on the overflow.

Figure 18-9. Repetition timechart for center-aligned counter

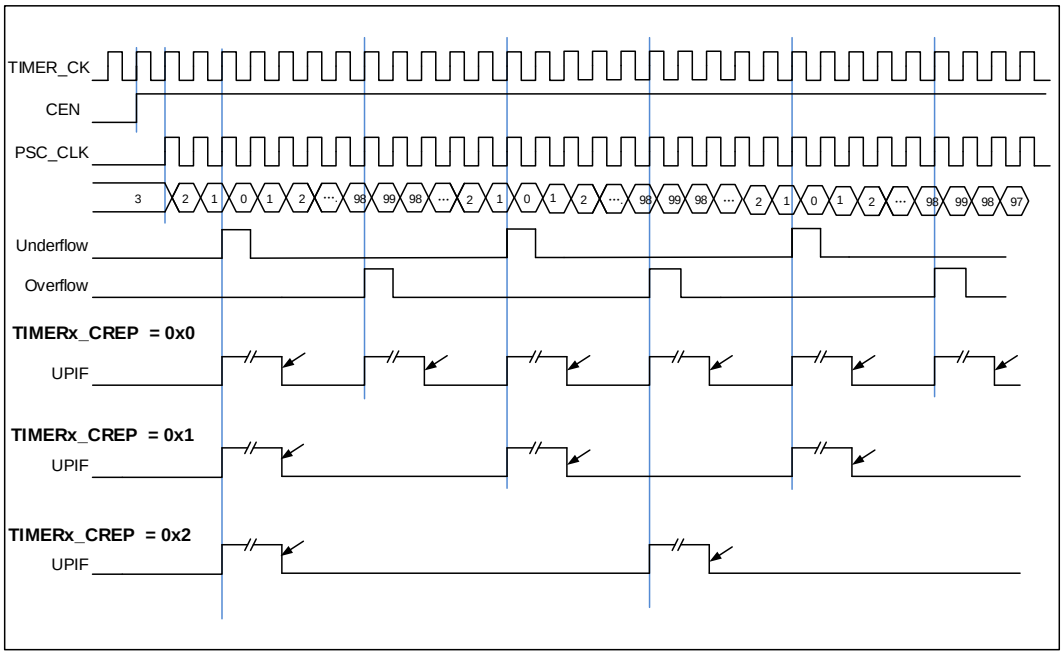


Figure 18-10. Repetition timechart for up-counter

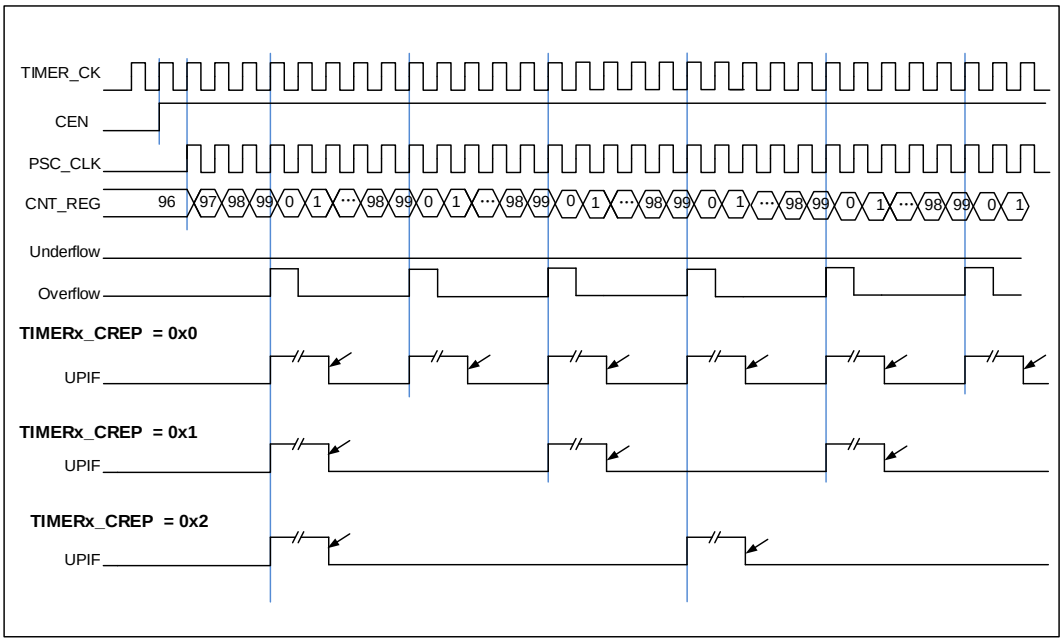
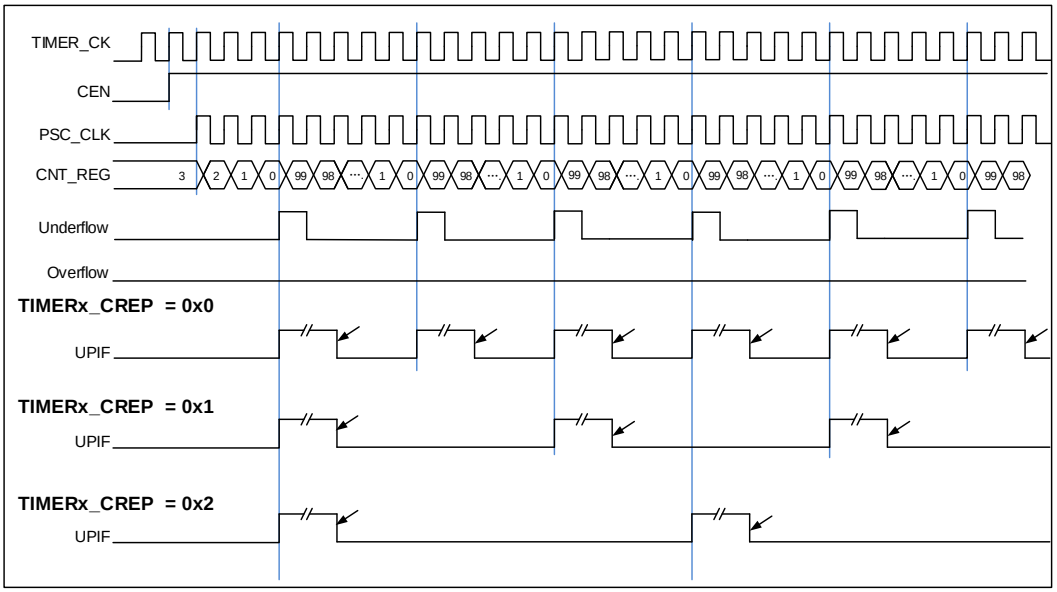


Figure 18-11. Repetition timechart for down-counter



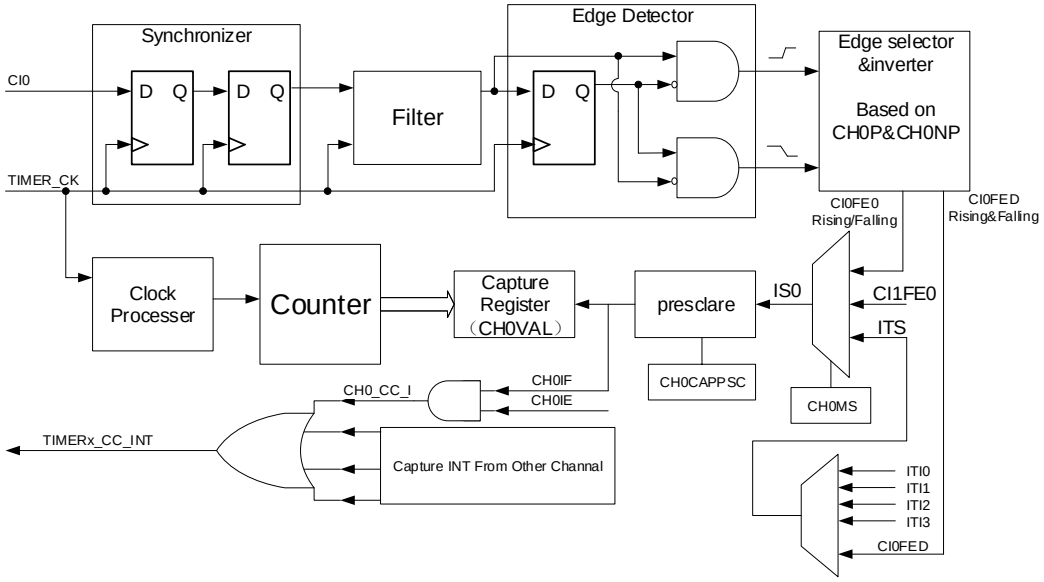
Input capture and output compare channels

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.

- **Channel input capture function**

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 18-12. Channel Input capture principle



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, TIMERx_CHxCV 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 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 channel input capture function 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.

- **Channel Output compare function**

Figure 18-13. Channel output compare principle (with complementary output, x=0,1,2)

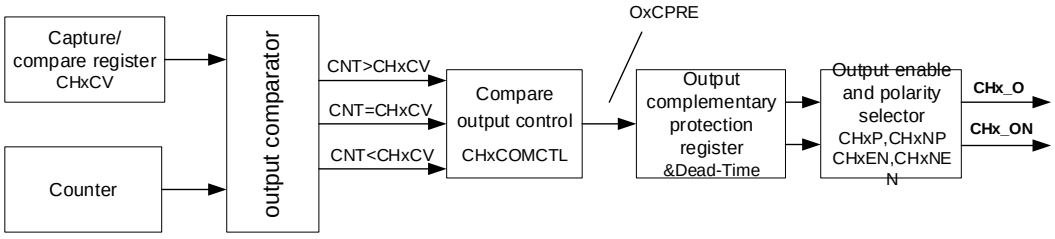


Figure 18-14. Channel output compare principle (CH3_O)

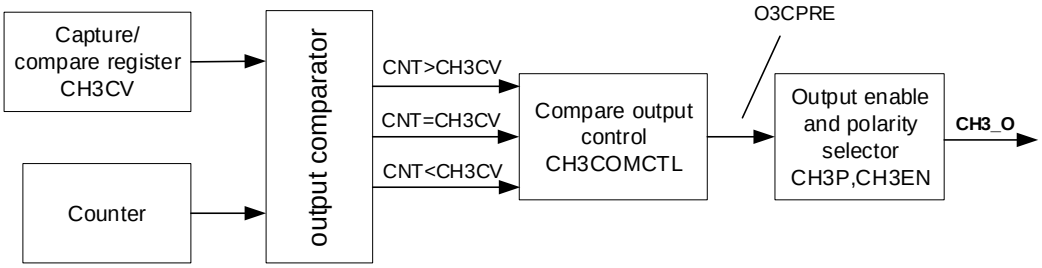


Figure 18-13. Channel output compare principle (with complementary output, x=0,1,2) and ***Figure 18-14. Channel output compare principle (CH3_O)*** show the principle circuit of channels output compare function. The relationship between the channel output signal CHx_O/CHx_ON and the OxCPRE signal (more details refer to ***Channel output prepare signal***) is described as blew: The active level of O0CPRE is high, the output level of CH0_O/CH0_ON depends on OxCPRE signal, CHxP/CHxNP bit and CHxEN/CHxNEN bit (please refer to the TIMERx_CHCTL2 register for more details). For examples,

1) Configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxEN=1

(the output of CHx_O is enabled),
If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;
If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

2) Configure CHxNP=0 (the active level of CHx_ON is low, contrary to OxCPRE), CHxNEN=1 (the output of CHx_ON is enabled),
If the output of OxCPRE is active(high) level, the output of CHx_O is active(low) level;
If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(high) level.

When CH0_O and CH0_ON are output at the same time, the specific outputs of CH0_O and CH0_ON are related to the relevant bits (ROS, IOS, POE and DTCFG bits) in the TIMERx_CCHP register. Please refer to ***Channel output complementary PWM*** for more details.

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 TIMERx_CHxCV 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 TIMERx_CHxCV register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be asserted, 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

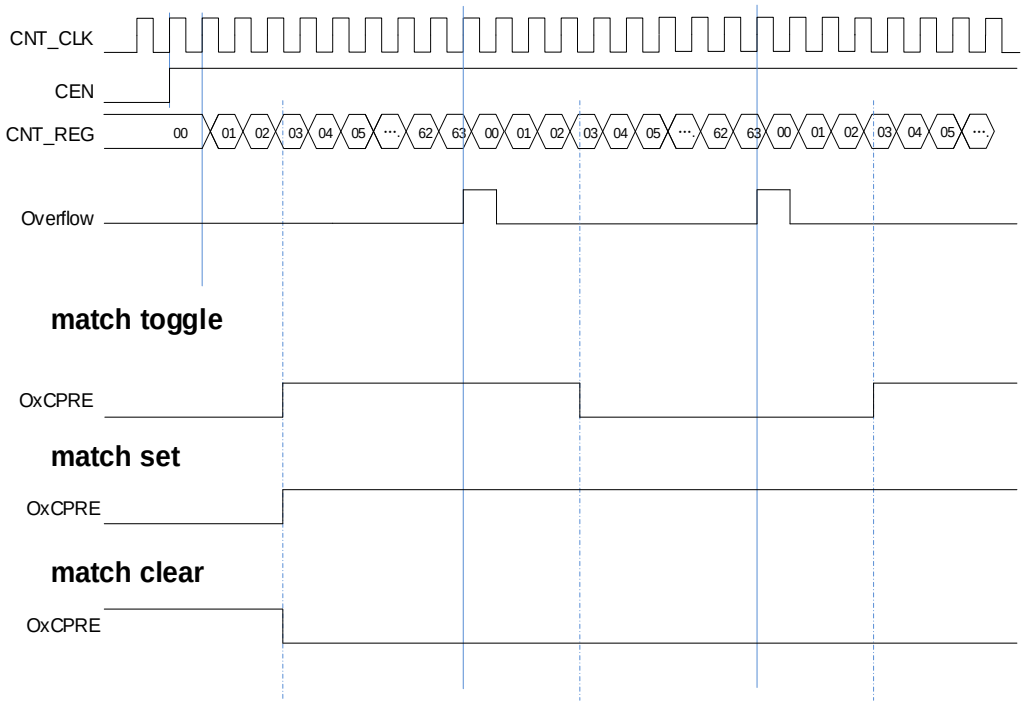
Step3: Interrupt/DMA-request enables configuration by CHxIE/CxCDE

Step4: Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV
About the TIMERx_CHxCV; you can change it on the go to meet the waveform you expected.

Step5: Start the counter by CEN.

Figure 18-15. Output-compare in three modes show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 18-15. Output-compare in three modes



Output PWM function

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 18-16. Timing chart of EAPWM*** 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 18-17. Timing chart of CAPWM*** 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 18-16. Timing chart of EAPWM

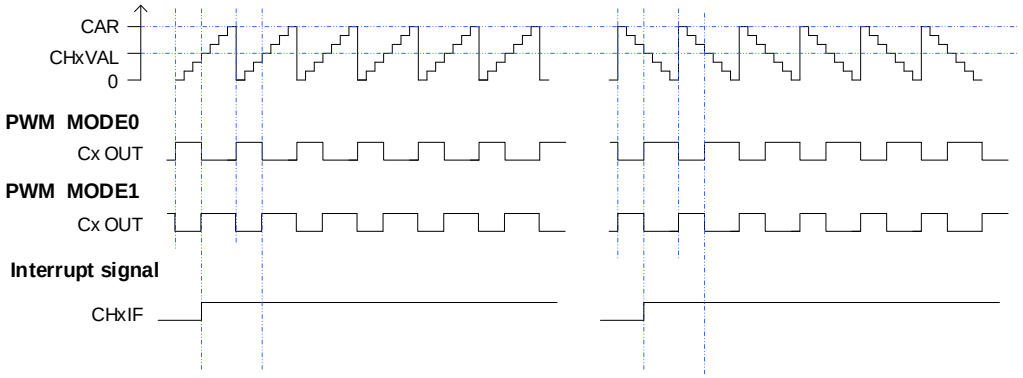
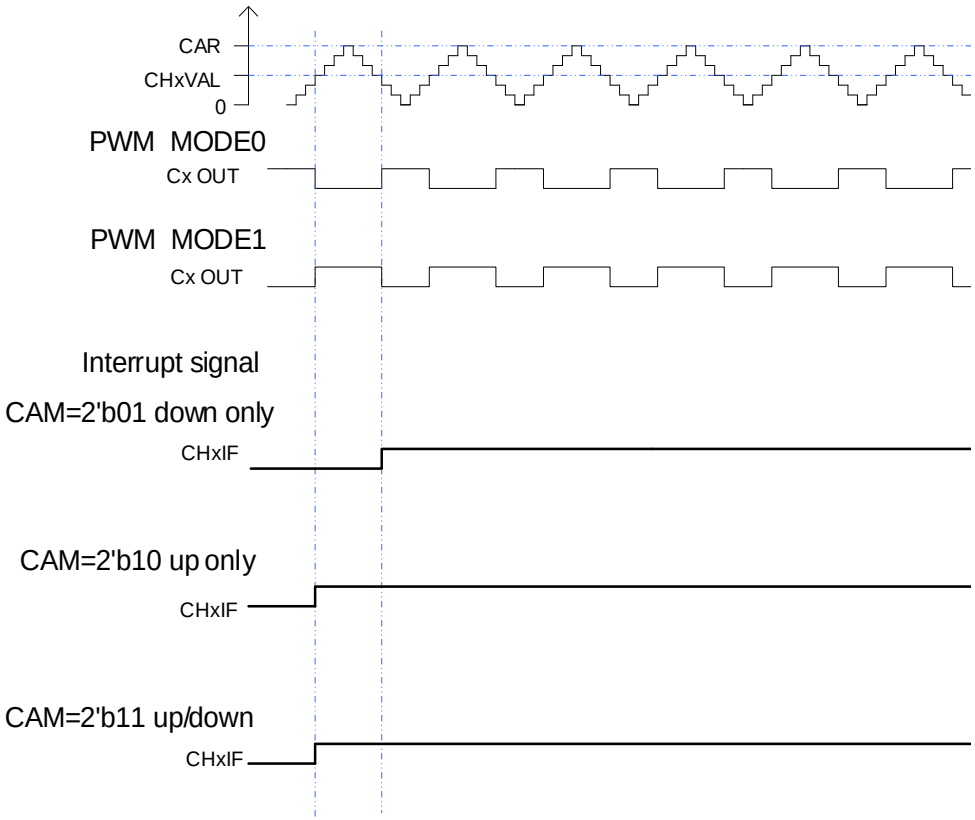


Figure 18-17. Timing chart of CAPWM



Channel output prepare signal

As is shown in ***Figure 18-13. Channel output compare principle (with complementary output, x=0,1,2)***,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 ETIFE 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.

Channel output complementary PWM

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 18-2. 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).	
				1	CHx_O/ CHx_ON output “off-state” (2): the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN. (3)	
			1	0		
				1		
		1	x	x	CHx_O/ CHx_ON output “off-state”: the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN.	
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⊕(4)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)(5)⊕CHxNP. CHx_ON output enable.
	1		0	0	CHx_O = CHxP CHx_O output “off-state”.	CHx_ON = CHxNP CHx_ON output “off-state”.
				1	CHx_O = CHxP CHx_O output “off-state”	CHx_ON =OxCPRE⊕CHxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON = CHxNP CHx_ON output “off-state”.
				1	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON =(!OxCPRE)⊕CHxNP CHx_ON output enable.

Note:

(1) output disable: the CHx_O / CHx_ON are disconnected to corresponding pins, the pin is floating with GPIO pull up/down setting which will be Hi-Z if no pull.

(2) “off-state”: CHx_O / CHx_ON output with inactive state (e.g., CHx_O = 0⊕CHxP = CHxP).

(3) See Break mode section for more details.

(4) ⊕: Xor calculate.

(5) (!OxCPRE)：the complementary output of the OxCPRE signal.

Insertion dead time for complementary PWM

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.

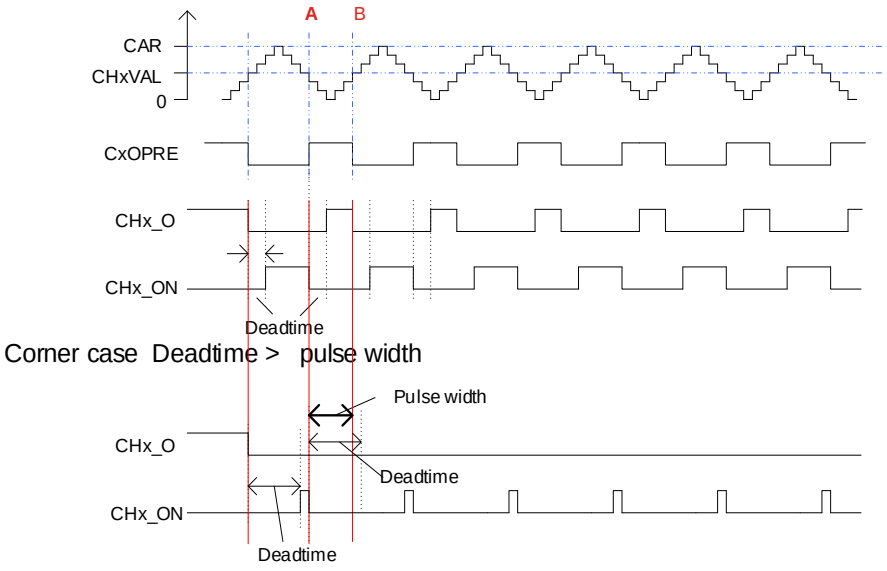
The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

When the channel (x) match (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled because under PWM0 mode. At point A in the ***Figure 18-18. 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 18-18. Complementary output with dead-time insertion.***)

Figure 18-18. Complementary output with dead-time insertion.



Break mode

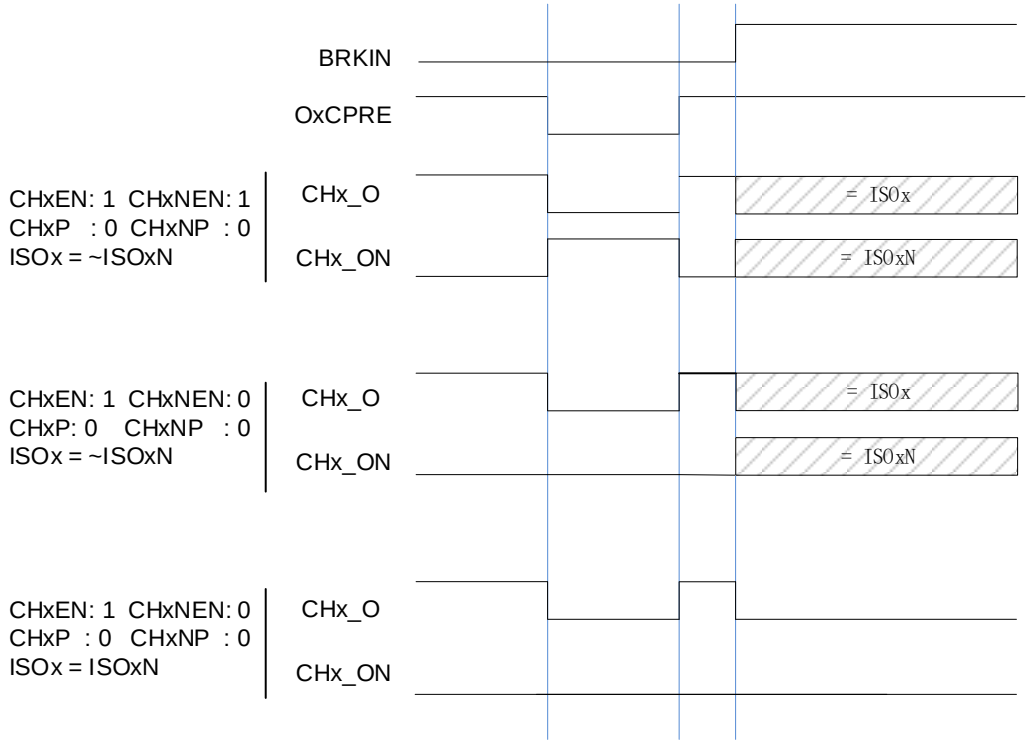
In this mode, the output CHx_O and CHx_ON are controlled by the POEN, IOS and ROS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register and cannot be set both to active level when break occurs. The break sources are input break pin and HXTAL stuck event by Clock Monitor (CKM) 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 18-19. Output behavior of the channel in response to a break (the break high active)



Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0FE0 and CI1FE1 derived from the TIMERx_CH0 and TIMERx_CH1 pins respectively to interact to control the counter value. The DIR bit is modified during each input source transition. The counter can be changed by the edges of CI0FE0 only, CI1FE1 only or both CI0FE0 and CI1FE1, the selection mode by setting the SMC[2:0] to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in ***Table 18-3. Counting direction in different quadrature decoder mode***. 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-period value. Therefore, TIMERx_CAR register must be configured before the counter starts to count.

Table 18-3. Counting direction in different quadrature decoder mode

Counting mode	Level	CI0FE0		CI1FE1	
		Rising	Falling	Rising	Falling
Quadrature decoder mode 0 SMC[2:0]=3'b000	CI1FE1=1	Down	Up	-	-
	CI1FE1=0	Up	Down	-	-
Quadrature decoder mode 1 SMC [2:0]=3'b010	CI0FE0=1	-	-	Up	Down
	CI0FE0=0	-	-	Down	Up
Quadrature decoder mode 2 SMC [2:0]=3'b011	CI1FE1=1	Down	Up	X	X
	CI1FE1=0	Up	Down	X	X
	CI0FE0=1	X	X	Up	Down
	CI0FE0=0	X	X	Down	Up

Note: "-" means "no counting"; "X" means impossible. "0" means "low level", "1" means "high level".

Figure 18-20. Counter behavior with CI0FE0 polarity non-inverted in mode 2

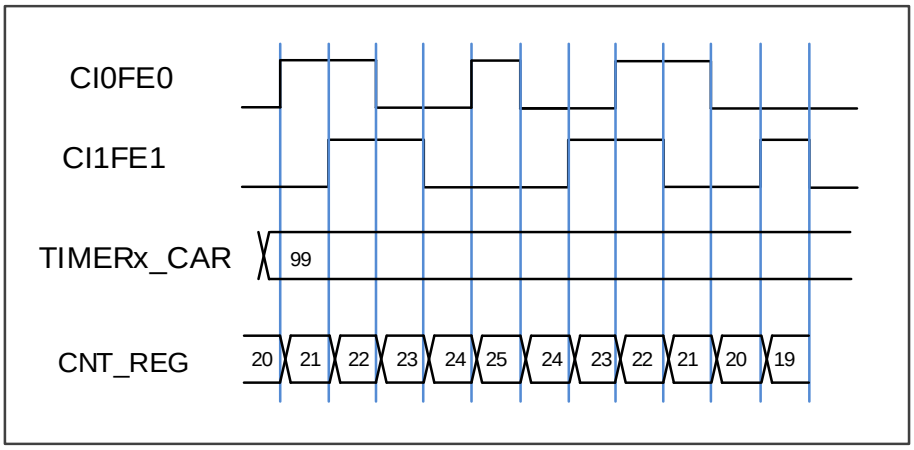
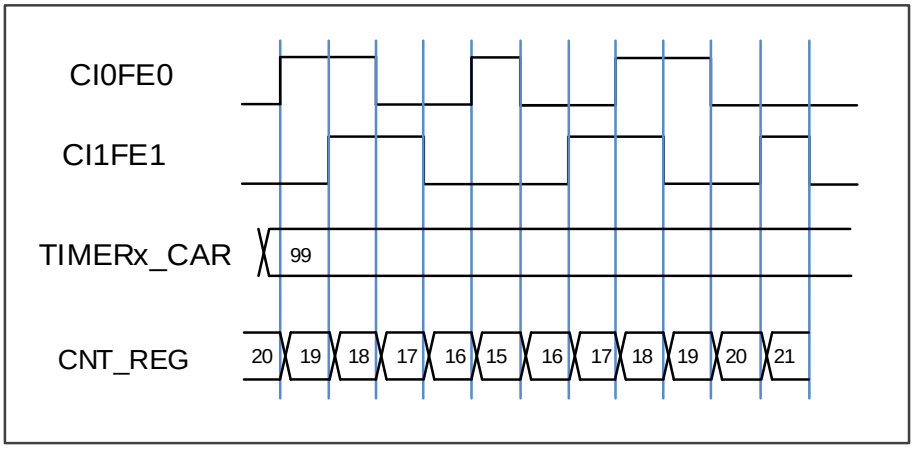


Figure 18-21. Counter behavior with CI0FE0 polarity inverted in mode 2



Hall sensor function

Hall sensor is generally used to control BLDC Motor; advanced timer can support this function.

Figure 18-22. 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 HALL sensor signals.

Each of the three input of HALL 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/General L0 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)

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 18-22. Hall sensor is used to BLDC motor

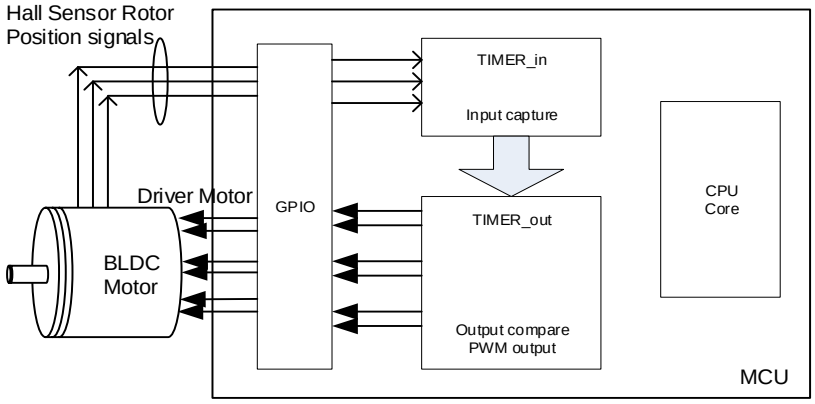
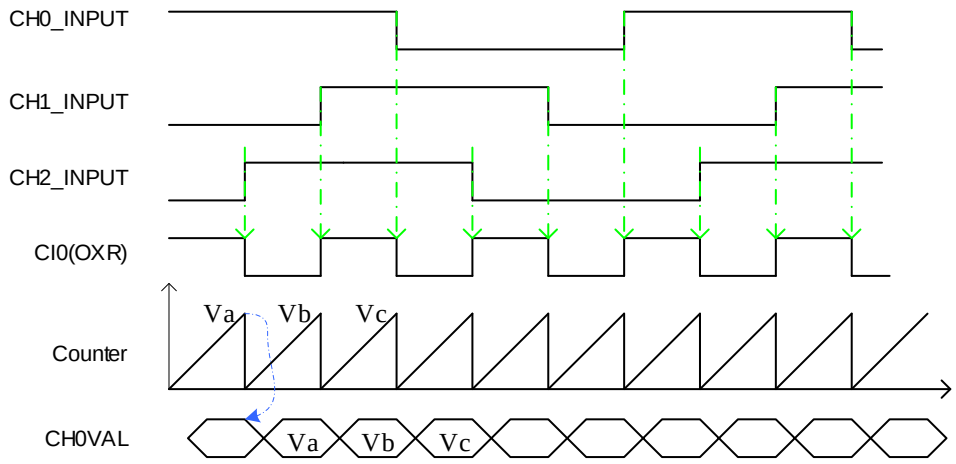
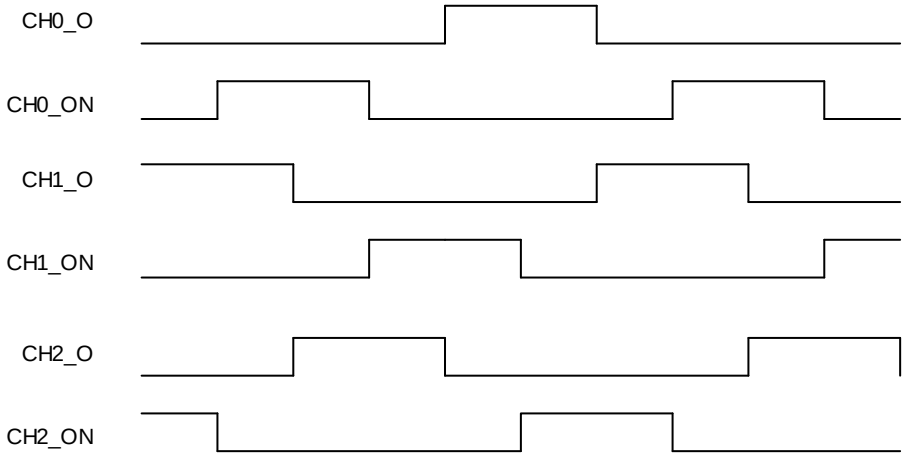


Figure 18-23. Hall sensor timing between two timers

Advanced/General L0 TIMER_in under input capture mode



Advanced TIMER_out under output compare mode(PWM with Dead-time)

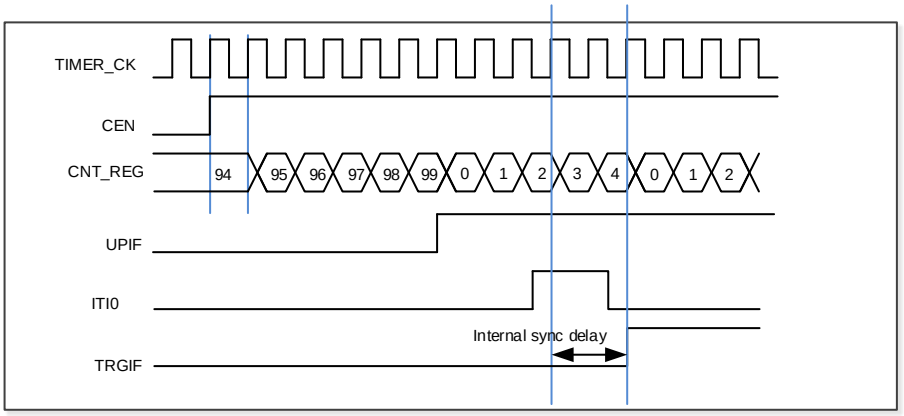


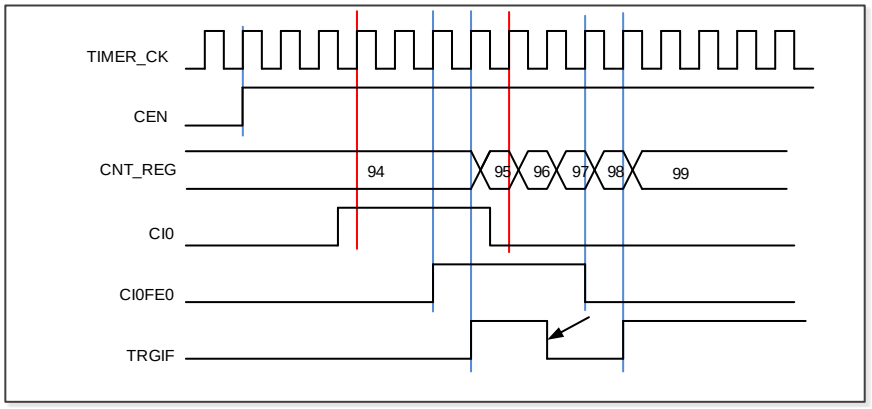
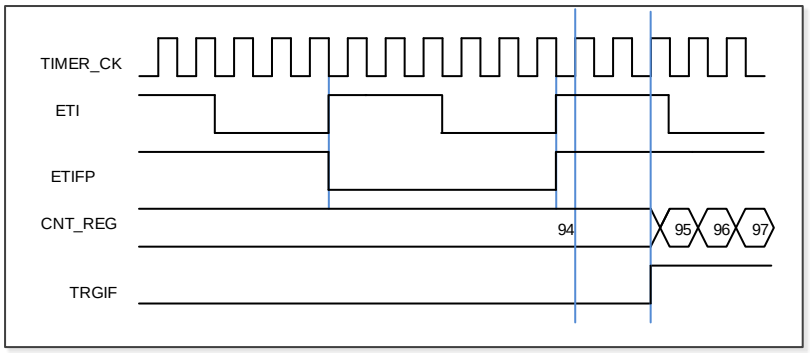
Master-slave management

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 18-4. Examples of slave mode

	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 CI0FE0 or CI1FE1 is selected as the trigger source, configure the CHxP and CHxNP for the polarity selection and inversion.	For the ITIx, no filter and prescaler can be used. For the CIx, filter can be used by configuring CHxCAPFLT, no

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
		101: CI0FE0 110: CI1FE1 111: ETIFP	If ETIFP is selected as the trigger source, configure the ETP for polarity selection and inversion.	prescaler can be used. For the ETIFP, filter can be used by configuring ETFC and prescaler can be used by configuring ETPSC.
Exam1	**Restart mode** The counter will be cleared and restart when a rising edge of trigger input comes.	TRGS[2:0] = 3'b000 ITI0 is selected.	For ITI0, no polarity selector can be used.	For the ITI0, no filter and prescaler can be used.
	Figure 18-24. Restart mode 			
Exam2	**Pause mode** The counter will be paused when the trigger input is low, and it will start when the trigger input is high.	TRGS[2:0]=3'b101 CI0FE0 is selected.	TI0S=0 (Non-xor) [CH0NP=0, CH0P=0] CI0FE0 does not invert. The capture event will occur on the rising edge only.	Filter is bypassed in this example.

<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 18-25. Pause mode
 
</td></tr>
<tr><td rowspan="2">Exam3</td><td>Event mode
The counter will start to count when a rising edge of trigger input comes.</td><td>TRGS[2:0] =3’b111
ETIFP is selected.</td><td>ETP = 0, the polarity of ETI does not change.</td><td>ETPSC = 1, ETI is divided by 2.
ETFC = 0, ETI does not filter.</td></tr>
<tr><td colspan="4">Figure 18-26. Event mode
 
</td></tr>
</table>

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. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

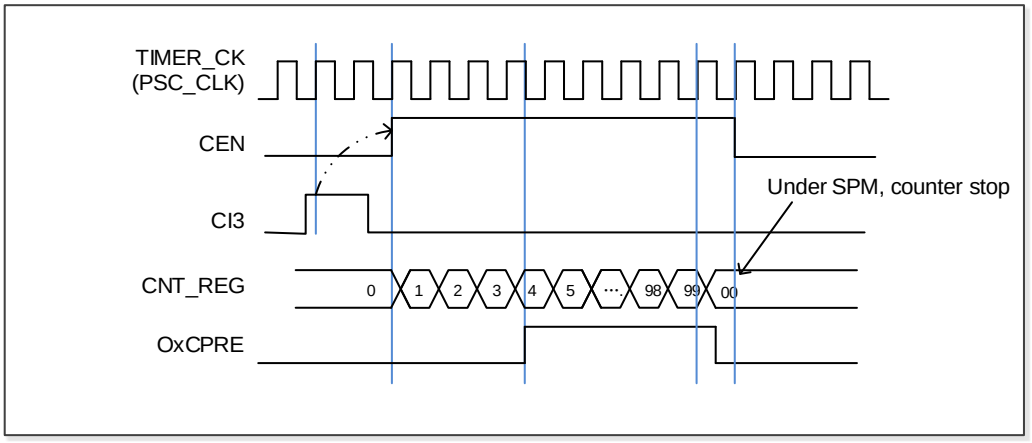
Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1 using software. Setting the 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.

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 18-27. Single pulse mode, TIMERx_CHxCV = 4, TIMERx_CAR=99 shows an example.

Figure 18-27. Single pulse mode, TIMERx_CHxCV = 4, TIMERx_CAR=99

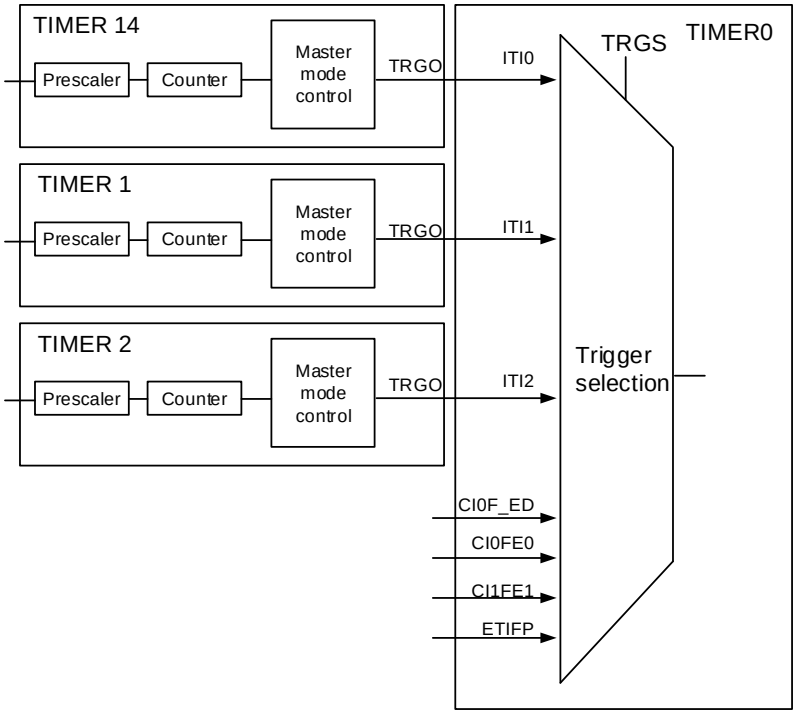


Timers interconnection

Timer can be configured as interconnection, that is, one timer which operate in the master mode outputs TRGO signal to control another timer which operate in the slave mode, TRGO include reset evevt, start evevt, update evevt, capture/compare pulse evevt, compare evevt. slave timer received the ITIx and performs the corresponding mode, include internal clock mode, quadrature decoder mode, restart mode, pause mode, event mode, external clock mode.

Figure 18-28. Timer0 master/slave mode example shows the timer0 trigger selection when it is configured in slave mode.

Figure 18-28. Timer0 master/slave mode example



Other interconnection examples:

- TIMER2 as the prescaler for TIMER0

TIMER2 is configured as a prescaler for TIMER0. Refer to ***Figure 18-28. Timer0 master/slave mode example for connections***. Steps are shown as follows:

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 TIMER2 period (TIMER2_CAR register).
3. Select TIMER2 as TIMER0 input trigger source (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' to the CEN bit (TIMER0_CTL0 register).
6. Start TIMER2 by writing '1' to the CEN bit (TIMER2_CTL0 register).

- Using an external trigger to start two timers synchronously.

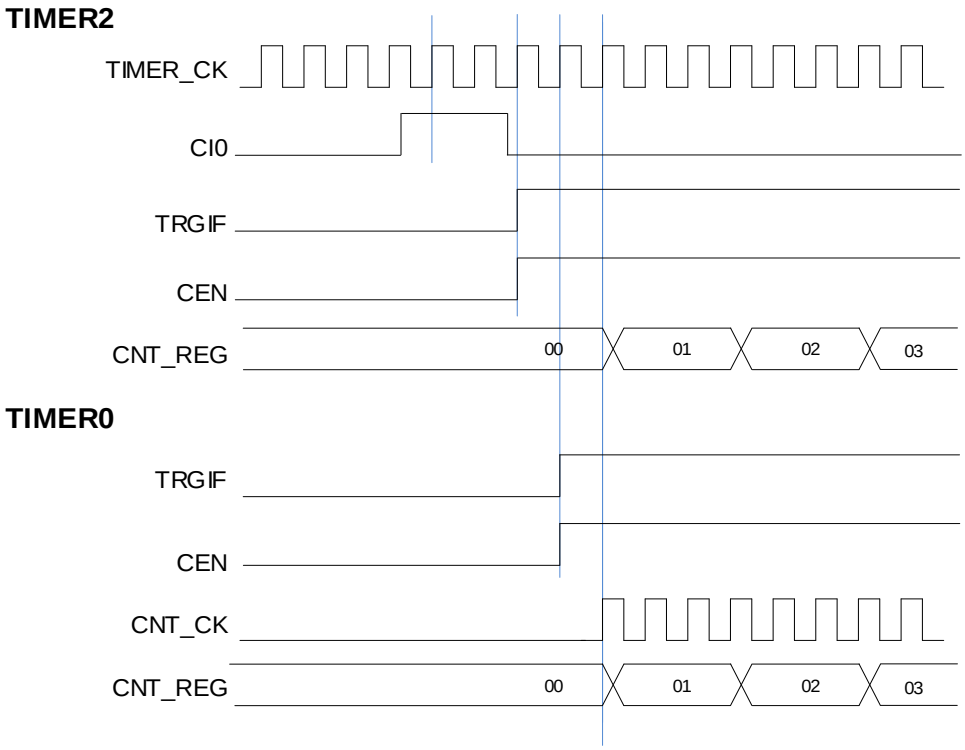
The start of TIMER0 is triggered by the enable signal of TIMER2, and TIMER2 is triggered by its CI0 input rising edge. To ensure that two timers start synchronously, TIMER2 must be configured in master/slave mode. Steps are shown as follows:

1. Configure TIMER2 in slave mode, and select CI0_ED as the input trigger (TRGS=3'b100 in the TIMER2_SMCFG register).
2. Configure TIMER2 in event mode (SMC=3'b110 in the TIMER2_SMCFG register).
3. Configure TIMER2 in master/slave mode by writing MSM=1 (TIMER2_SMCFG register).

4. Select TIMER2 as TIMER0 input trigger source (TRGS=3'b010 in the TIMERx_SMCFG register).
5. Configure TIMER0 in event mode (SMC=3'b110 in the TIMER0_SMCFG register).

When the CI0 signal of TIMER2 generates a rising edge, two timer counters start counting synchronously with the internal clock and both TRGIF flags are set.

Figure 18-29. Trigger TIMER0 and TIMER2 by the CI0 signal of TIMER2



Timer DMA mode

Timer DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Corresponding DMA request bit should be asserted to enable DMA request for internal interrupt event. TIMERx will send a request to DMA when the interrupt event occurs. DMA is configured to M2P (memory to peripheral) mode and the address of TIMERx_DMATB is configured to PADDR (peripheral base address), then DMA will access the TIMERx_DMATB. In fact, TIMERx_DMATB register 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), the timer sends only one DMA request. 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 and DMATA+0xC at the next 3 accesses to TIMERx_DMATB. In a word, one-time DMA internal interrupt event asserts, (DMATC+1) times request will be sent by TIMERx.

If one more DMA request event occurs, TIMERx will repeat the process above.

Timer debug mode

When the Cortex®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL0 register is set to 1, the TIMERx counter stops.

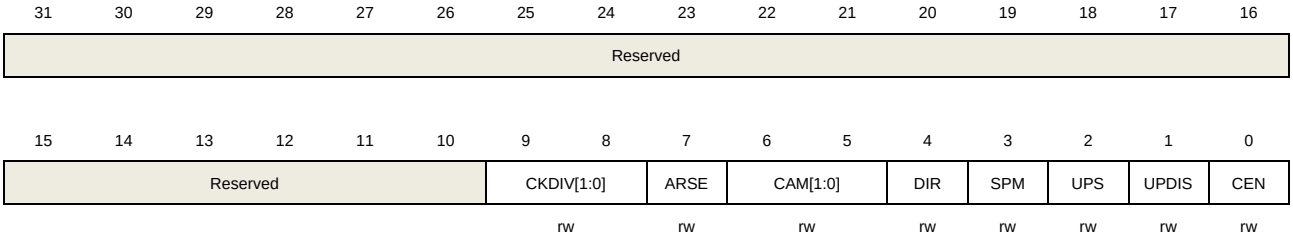
18.1.5. TIMERx registers(x=0, 7)

TIMER0 base address: 0x4001 2C00
TIMER7 base address: 0x4001 3400

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 factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK\_TIMER}$ 01: $f_{DTS}= f_{CK\_TIMER} /2$ 10: $f_{DTS}= f_{CK\_TIMER} /4$ 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 counting down, CHxF bit 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 counting up, CHxF bit 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 counting up and counting down, CHxF bit

		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 If the timer work in center-aligned mode or quadrature decode mode, this bit is read only.
3	SPM	Single pulse mode. 0: Single pulse mode disable. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: Update event disable. **Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.
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 quadrature decode mode.

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: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source: Master timer generate a reset the UPG bit in the TIMERx_SWEVG register is set

		001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source : CEN control bit is set The trigger input in pause mode is high 010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit. 011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output. 100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE. 101: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O1CPRE. 110: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O2CPRE. 111: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O3CPRE.
3	DMAS	DMA request source selection 0: When capture or compare event occurs, the DMA request of channel x is sent 1: When update event occurs, the DMA request of channel x is sent.
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 rising edge or high level . 1: ETI is active at falling edge or low level .
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. When the slave mode is configured as restart mode, pause mode or event mode, the timer can still work in the external clock 1 mode by setting this bit. But the TRGS bits must not be 3'b111 in this case. The clock source of the timer will be ETIFP if external clock mode 0 and external clock mode 1 are configured at the same time. **Note:** External clock mode 0 enable is in this register's SMC[2:0] bit-filed.
13:12	ETPSC[1:0]	The prescaler of external trigger 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: The prescaler is 2. 10: The prescaler is 4. 11: The prescaler is 8.
11:8	ETFC[3:0]	External trigger filter control The external trigger can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the external trigger signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit-field, it is considered to be an effective level. The filtering capability configuration is as follows:

EXTFC[3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{TIMER_CK}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS\_CK}/2$
4'b0101	8	
4'b0110	6	$f_{DTS\_CK}/4$
4'b0111	8	
4'b1000	6	$f_{DTS\_CK}/8$
4'b1001	8	
4'b1010	5	$f_{DTS\_CK}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS\_CK}/32$
4'b1110	6	
4'b1111	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: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: 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 CI0FE0 edge, while the direction depends on CI1FE1 level.

010: Quadrature decoder mode 1.The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 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 an update event is generated 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 clock when it is low.
110: Event mode. A rising edge of the trigger input enables the counter.
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

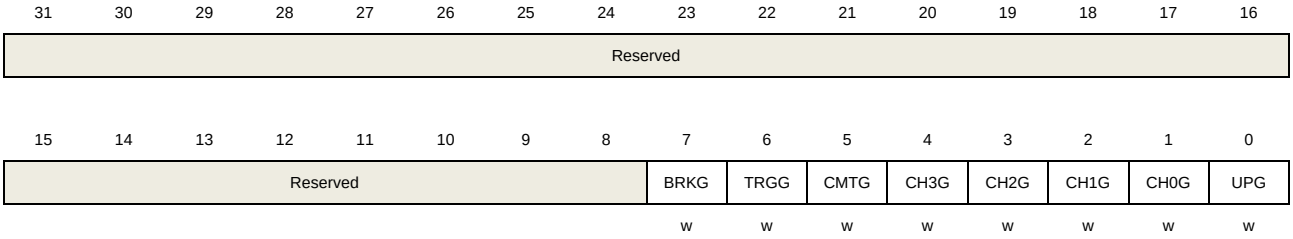
Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12	CH3OF	Channel 3 over capture flag Refer to CH0OF description
11	CH2OF	Channel 2 over capture flag Refer to CH0OF description
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag When 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 When the break input is inactive, the bit is set by hardware. When the break input is inactive, the bit can be cleared by software. 0: No active level break has been detected. 1: An active level has been detected.
6	TRGIF	Trigger interrupt flag This flag is set on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can 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. If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 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).



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).

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 CH0COMFEN 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 programmed as output mode 01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1 10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1 11: Channel 1 is programmed as input mode, IS1 is connected to ITS. **Note:** When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	Channel 0 output compare clear enable. When this bit is set, if the ETIFP signal is detected as high level, the O0CPRE signal will be cleared. 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 specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity 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 is

		equals to the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced to low level. 101: Force high. O0CPRE is forced to high level. 110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise. 111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise. If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result 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 verifying the shadow register only in single pulse mode (when SPM=1) 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. 1: Channel 0 output quickly compare enable.
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 programmed as output mode 01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0 10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0

11: Channel 0 is programmed as input mode, IS0 is connected to ITS

Note: When CH0MS[1:0]=11, it is necessary to select an internal trigger input 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 The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI0 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH0CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	
4'b1111	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, input capture occurs on every channel input edge 01: The input capture occurs on every 2 channel input edges 10: The input capture occurs on every 4 channel input edges 11: The input capture occurs on 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 CH0COMFEN 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 programmed as output mode 01: Channel 3 is programmed as input mode, IS3 is connected to CI3FE3 10: Channel 3 is programmed as input mode, IS3 is connected to CI2FE3 11: Channel 3 is programmed as input mode, IS3 is connected to ITS.

		Note: When CH3MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH2COMCEN	Channel 2 output compare clear enable. When this bit is set, if the ETIFP signal is detected as high level, the O2CPRE signal will be cleared. 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 specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP 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 is equals to the output compare register TIMERx_CH2CV. 010: Clear the channel output. O2CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH2CV. 011: Toggle on match. O2CPRE toggles when the counter is equals to the output compare register TIMERx_CH2CV. 100: Force low. O2CPRE is forced to low level. 101: Force high. O2CPRE is forced to high level. 110: PWM mode 0. When counting up, O2CPRE is high when the counter is smaller than TIMERx_CH2CV, and low otherwise. When counting down, O2CPRE is low when the counter is larger than TIMERx_CH2CV, and high otherwise. 111: PWM mode 1. When counting up, O2CPRE is low when the counter is smaller than TIMERx_CH2CV, and high otherwise. When counting down, O2CPRE is high when the counter is larger than TIMERx_CH2CV, and low otherwise. If configured in PWM mode, the O2CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result 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 verifying the shadow register only in single pulse mode (when SPM=1) 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. 1: Channel 2 output quickly compare enable.
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 programmed as output mode 01: Channel 2 is programmed as input mode, IS2 is connected to CI2FE2 10: Channel 2 is programmed as input mode, IS2 is connected to CI3FE2 11: Channel 2 is programmed as input mode, IS2 is connected to ITS. **Note:** When CH2MS[1:0]=11, it is necessary to select an internal trigger input 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 The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI2 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH2CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	

4’b0100	6	f_{DTS}/2
4’b0101	8	
4’b0110	6	f_{DTS}/4
4’b0111	8	
4’b1000	6	f_{DTS}/8
4’b1001	8	
4’b1010	5	f_{DTS}/16
4’b1011	6	
4’b1100	8	
4’b1101	5	f_{DTS}/32
4’b1110	6	
4’b1111	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, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on 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 complementary output high level is active level 1: Channel 0 complementary output low level is active level When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
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 high level is active level 1: Channel 0 low level is active level When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or

		trigger operation in slave mode. And CIxFE0 will not be inverted. [CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: Reserved. 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 TIMER_CK 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.

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 The bit can be set to 1 by: - Write 1 to this bit - If OAEN is set to 1, this bit is set to 1 at the next update event. The bit can be cleared to 0 by: - Write 0 to this bit - Valid fault input (asynchronous). 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: Disable channel outputs (CHxO or CHxON). 1: Enabled channel outputs (CHxO or CHxON). **Note:** This bit is only valid when CHxMS=2'b00.
14	OAEN	Output automatic enable 0: The POEN bit can only be set by software. 1: POEN can be set 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 input polarity
This bit specifies the polarity of the BRKIN input signal.
0: BRKIN input active low
1; BRKIN input active high

12 BRKEN

Break input 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” enable
When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.
0: “off-state” disabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is output disabled.
1: “off-state” enabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is “off-state”.
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” enable
When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.
0: “off-state” disabled. If the CHxEN/CHxNEN bits are both reset, the channels are output disabled.
1: “off-state” enabled. No matter the CHxEN/CHxNEN bits, the channels are “off-state”.
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

The relationship between DTVAL value and the duration of dead-time is as follow:

DTCFG[7:5]	The duration of dead-time
3'b0xx	DTCFG[7:0] * t_{DTS_CK}
3'b10x	(64+ DTCFG[5:0]) * t_{DTS_CK} *2
3'b110	(32+ DTCFG[4:0]) * t_{DTS_CK} *8
3'b111	(32+ DTCFG[4:0]) * t_{DTS_CK} *16

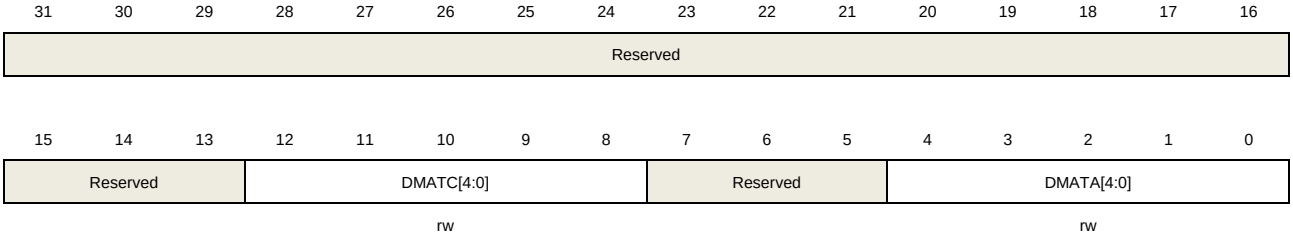
Note:

1. t_{DTS_CK} is the period of DTS_CK which is configured by CKDIV[1:0] in TIMERx_CTL0.
2. 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).



Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This filed defines the number(n) of the register that DMA will access(R/W), n = (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001.
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.

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.

Configuration register (TIMERx_CFG)

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
Reserved															

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

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CHVSEL	Write CHxVAL register selection This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	OUTSEL	The output value selection This bit-field set and reset by software

1: If POEN and IOS is 0, the output disabled

0: No effect

18.2. General level0 timer (TIMERx, x=1, 2, 3, 4)

18.2.1. 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(TIMER2~4) or 32-bit(TIMER1) 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.

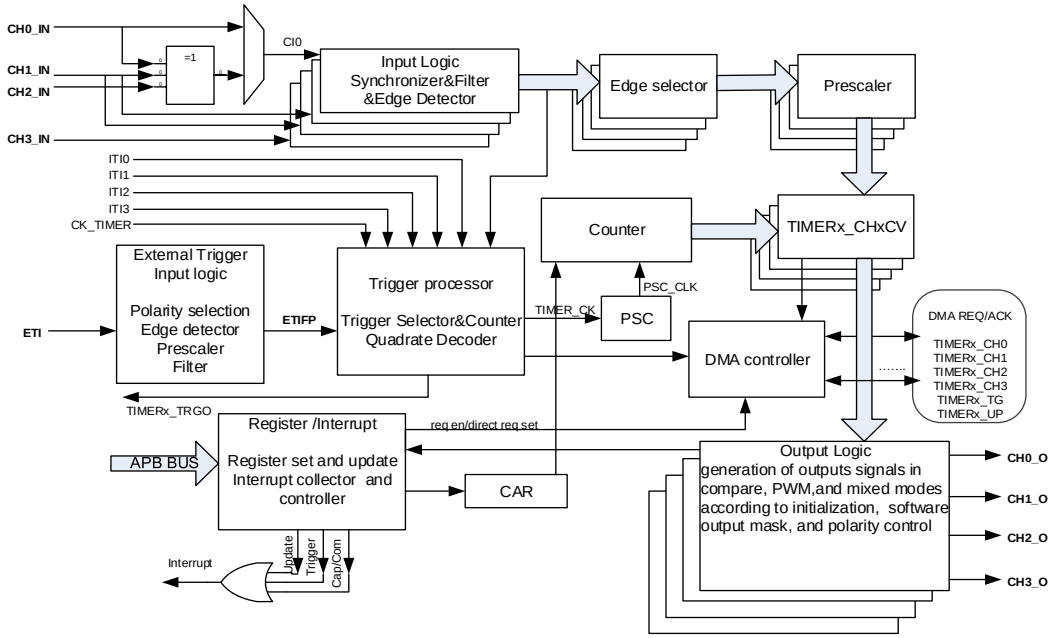
18.2.2. Characteristics

- Total channel num: 4.
- Counter width: 16bit(TIMER2~4),32bit(TIMER1).
- 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 management.

18.2.3. Block diagram

Figure 18-30. General Level 0 timer block diagram provides details on the internal configuration of the general level0 timer.

Figure 18-30. General Level 0 timer block diagram



18.2.4. Function overview

Clock source configuration

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]).

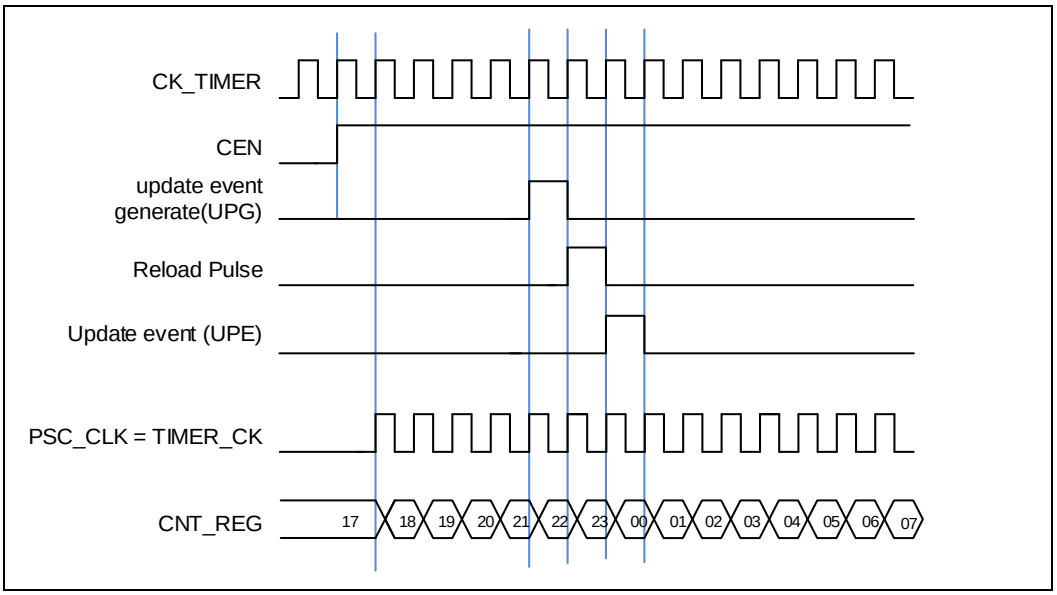
- SMC [2:0] == 3'b000. Internal timer clock CK_TIMER which is from module RCU.

The default internal clock source is the CK_TIMER used to drive the counter prescaler when the 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 SMC [2:0] in the TIMERx_SMCFG register are setting 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 SMC [2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK_TIMER is the counter prescaler driving clock source.

Figure 18-31. Timing chart of internal clock divided by 1



- SMC [2:0] == 3'b111(external clock mode 0). External input pin 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_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.

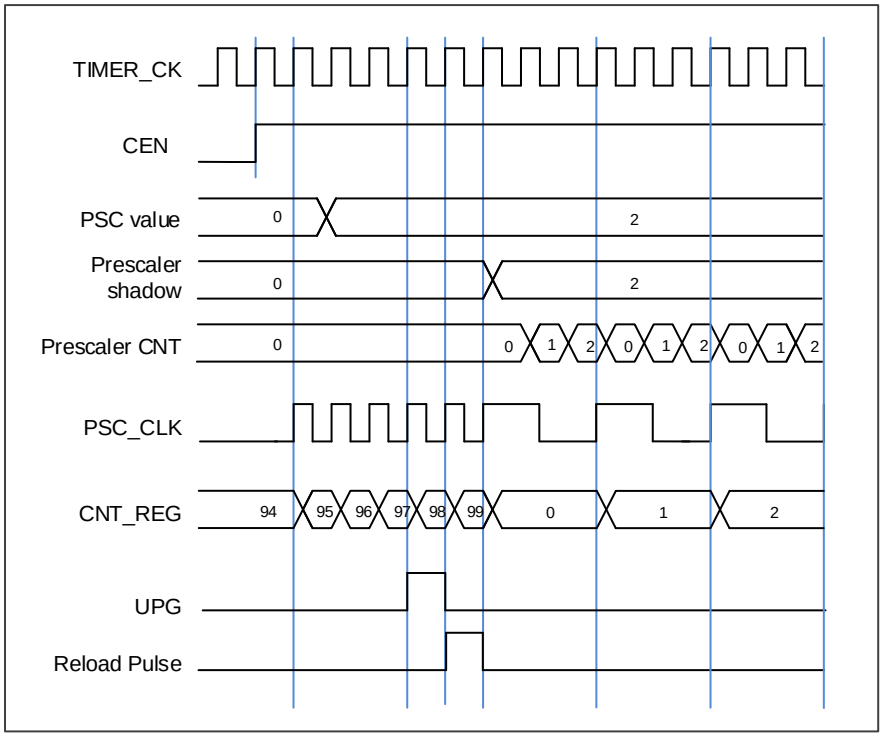
- SMC1== 1'b1(external clock mode 1). External input pin source (ETI)

The TIMER_CK, 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.

Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 18-32. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again. 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 shadow registers (counter autoreload register, prescaler register) are updated.

Figure 18-33. Timing chart of up counting mode, PSC=0/2 and ***Figure 18-34. Timing chart of up counting mode, change TIMERx_CAR ongoing*** show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

Figure 18-33. Timing chart of up counting mode, PSC=0/2

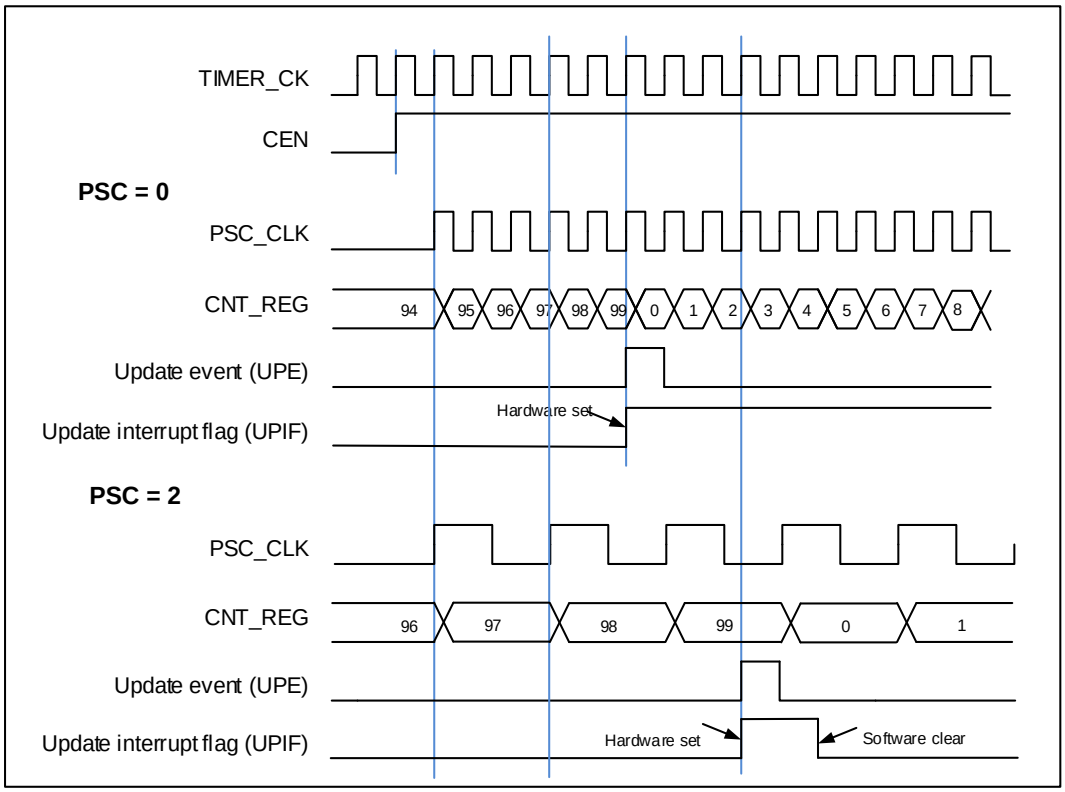
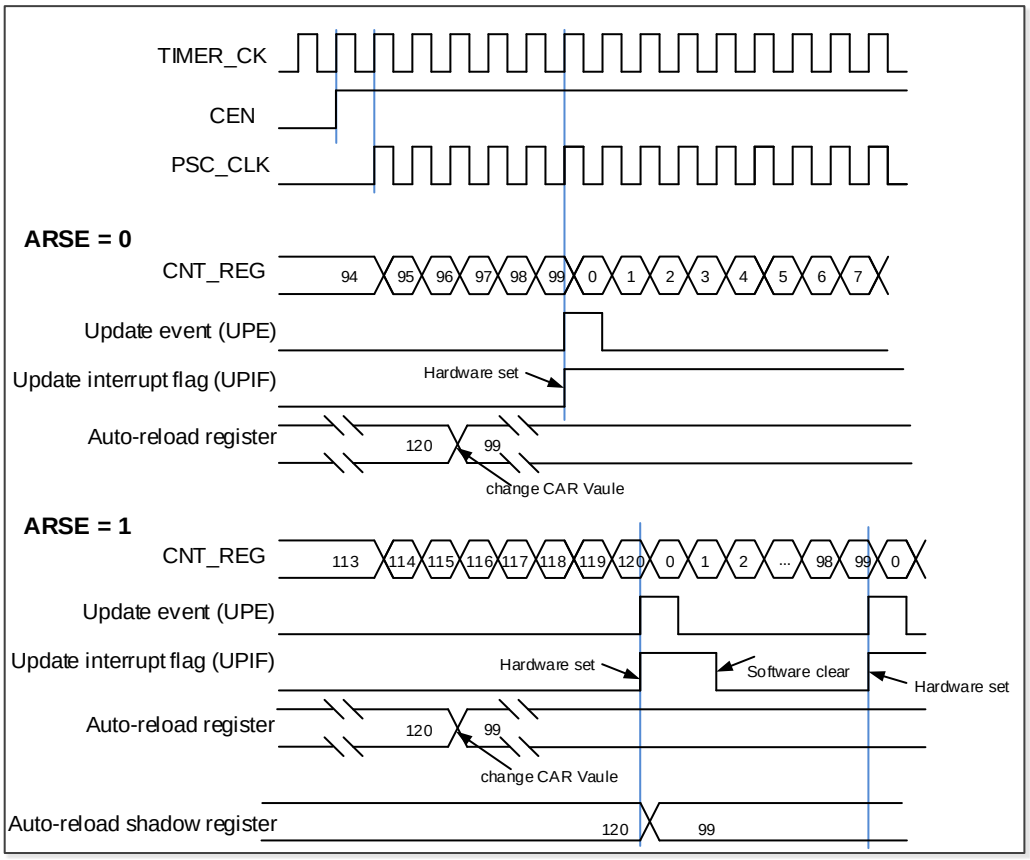


Figure 18-34. Timing chart of up counting mode, change TIMERx_CAR ongoing



Counter down counting

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 will start counting down from the counter-reload value. 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 shadow registers (counter autoreload register, prescaler register) are updated.

Figure 18-35. Timing chart of down counting mode,PSC=0/2 and *Figure 18-36. Timing chart of down counting mode, change TIMERx_CAR ongoing* show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x99.

Figure 18-35. Timing chart of down counting mode,PSC=0/2

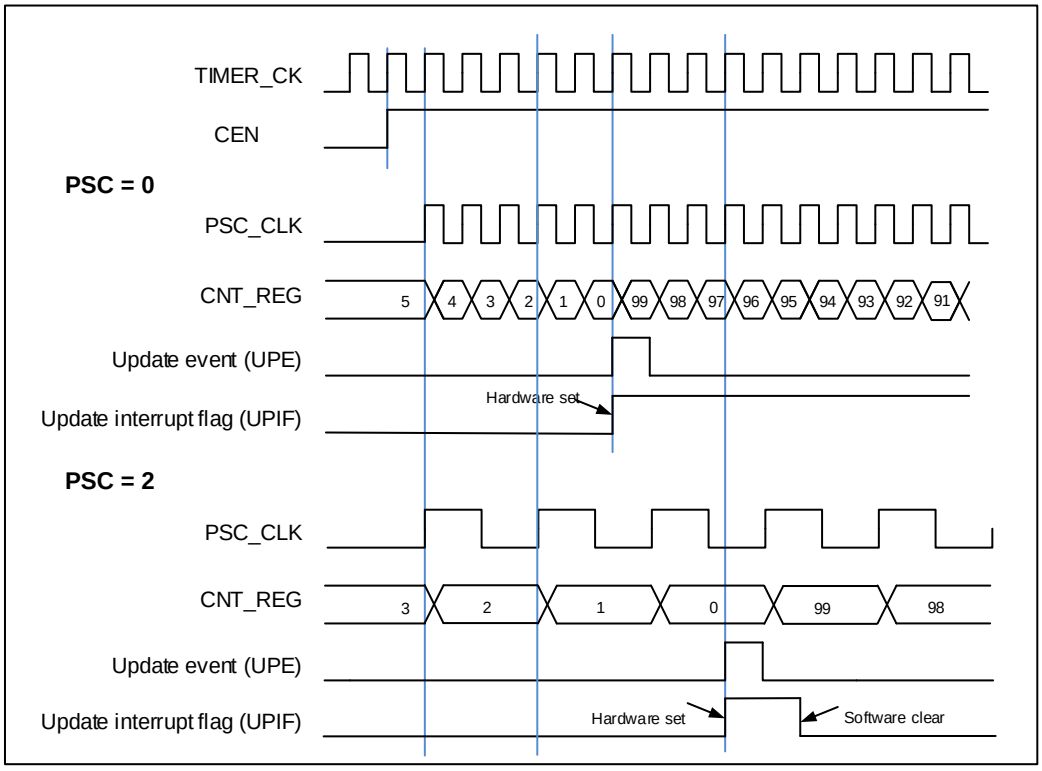
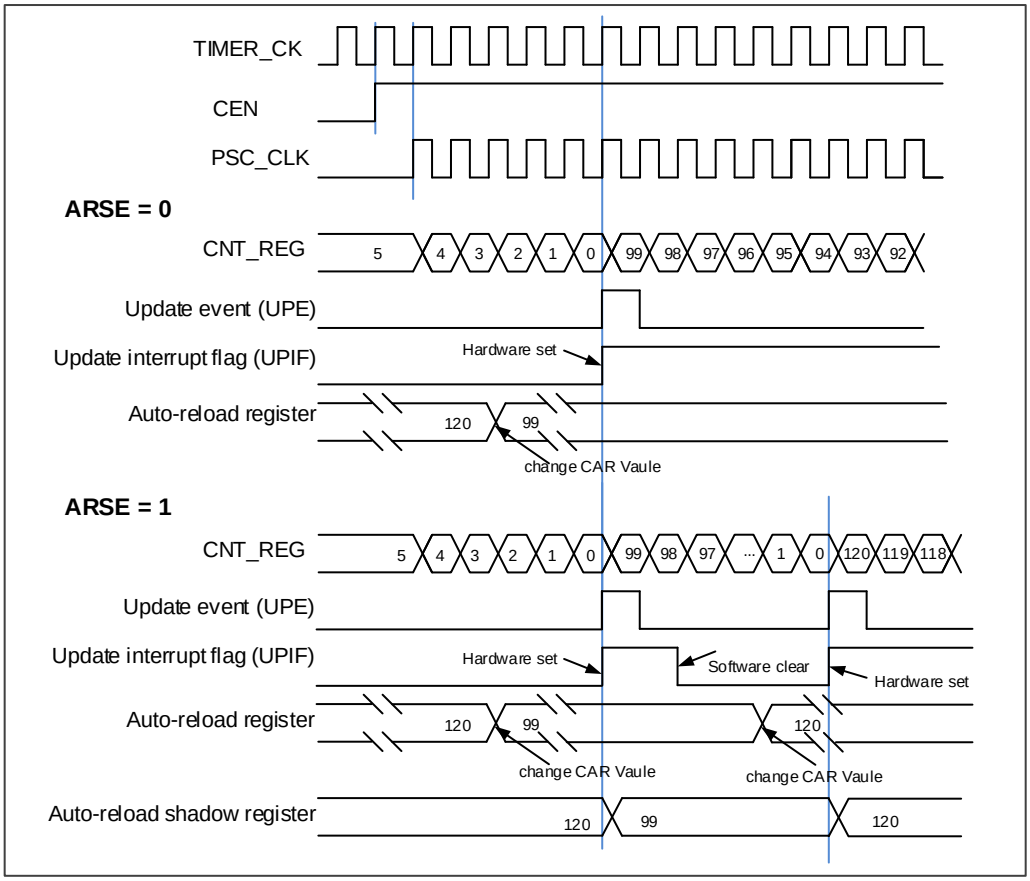


Figure 18-36. Timing chart of down counting mode, change TIMERx_CAR ongoing



Counter center-aligned counting

In this 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 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.

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.

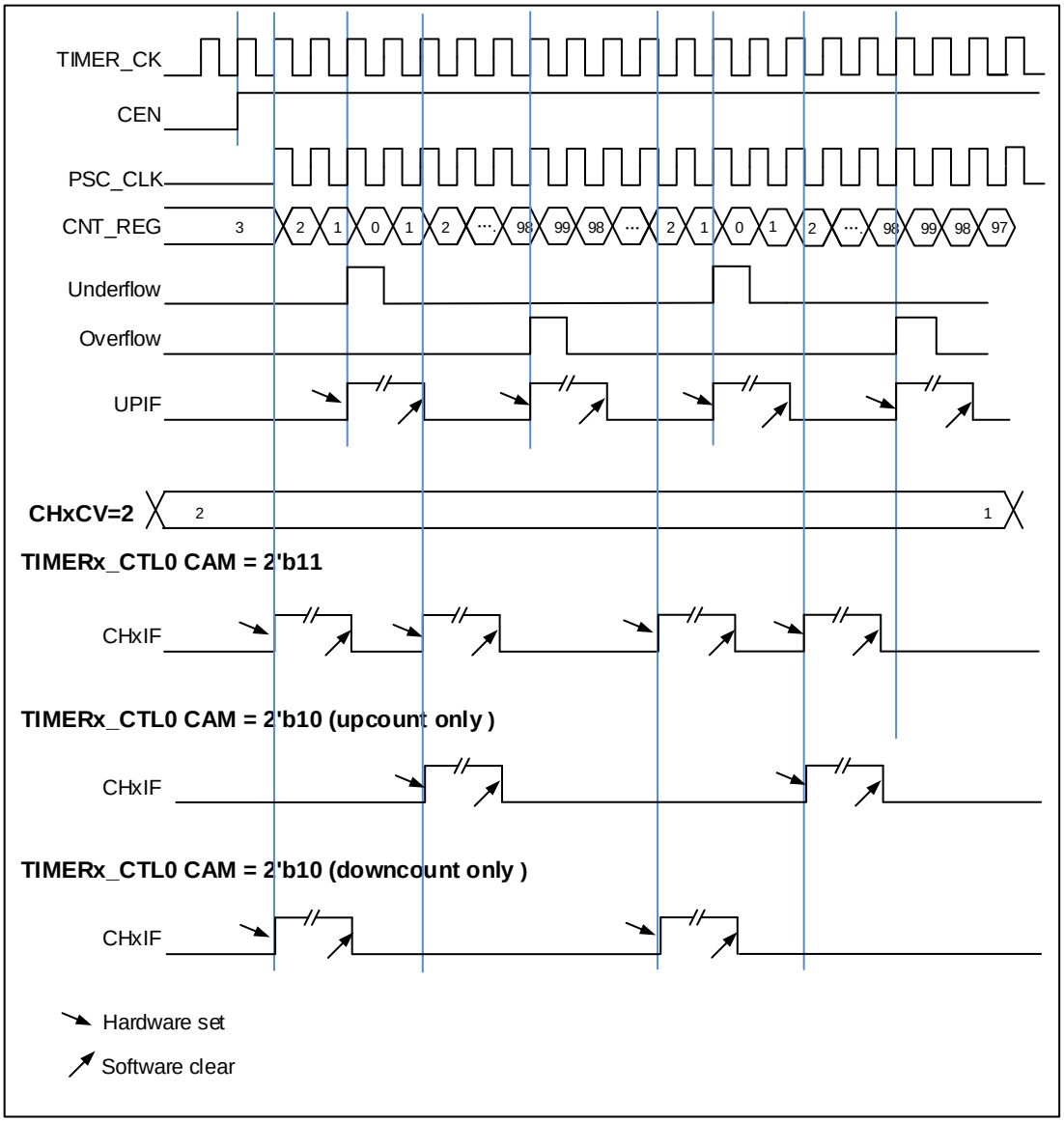
The UPIF bit in the TIMERx_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 18-37. Timing chart of center-aligned counting mode***

If the UPDIS bit in the TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter autoreload register, prescaler register) are updated.

Figure 18-37. Timing chart of center-aligned counting mode show some examples of the counter behavior when TIMERx_CAR=0x99. TIMERx_PSC=0x0

Figure 18-37. Timing chart of center-aligned counting mode



Input capture and output compare channels

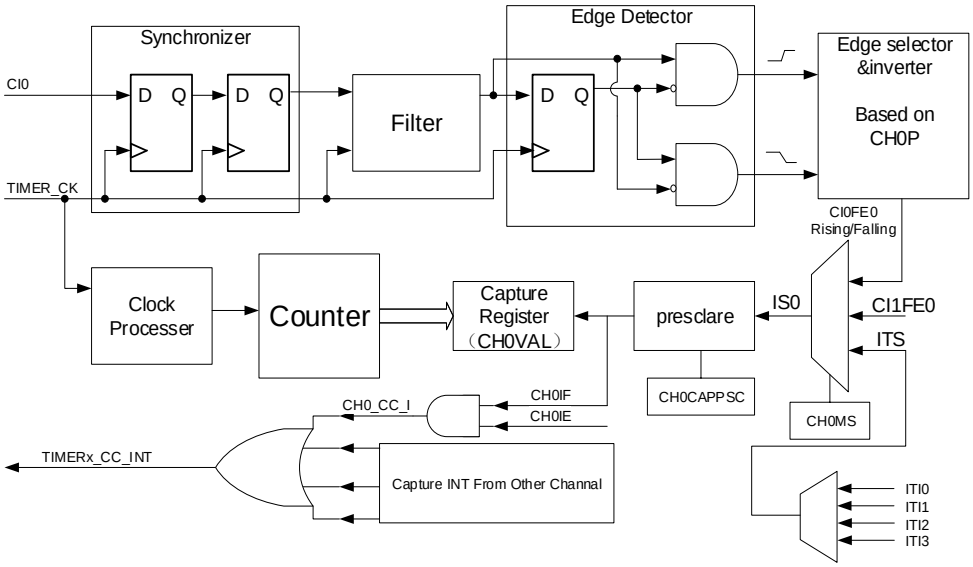
The general level0 Timer has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

- Channel input capture function

Channel input capture function 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 18-38. Channel input capture principle



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, TIMERx_CHxCV 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 in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP.

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 channel input capture function 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.

- Channel output compare function

Figure 18-39. Channel output compare principle (x=0,1,2,3)

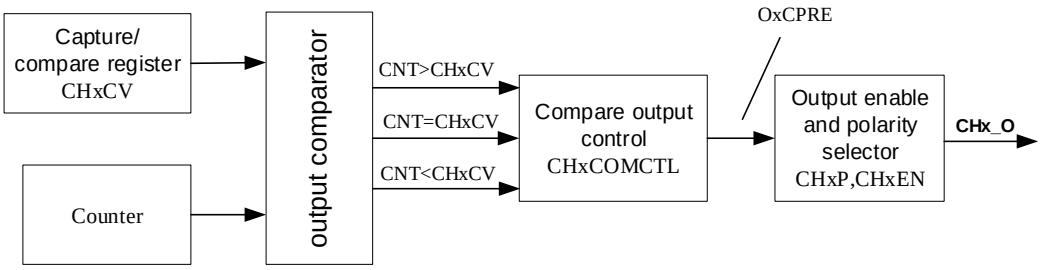


Figure 18-39. Channel output compare principle (x=0,1,2,3) shows the principle circuit of channels output compare function. The relationship between the channel output signal CHx_O and the OxCPRE signal (more details refer to *Channel output prepare signal*) is described as blew: The active level of O0CPRE is high, the output level of CH0_O depends on OxCPRE signal, CHxP bit and CH0P bit (please refer to the TIMERx_CHCTL2 register for more details).For example, configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxEN=1 (the output of CHx_O is enabled),

If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;

If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

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
* Enable the output by CHxEN

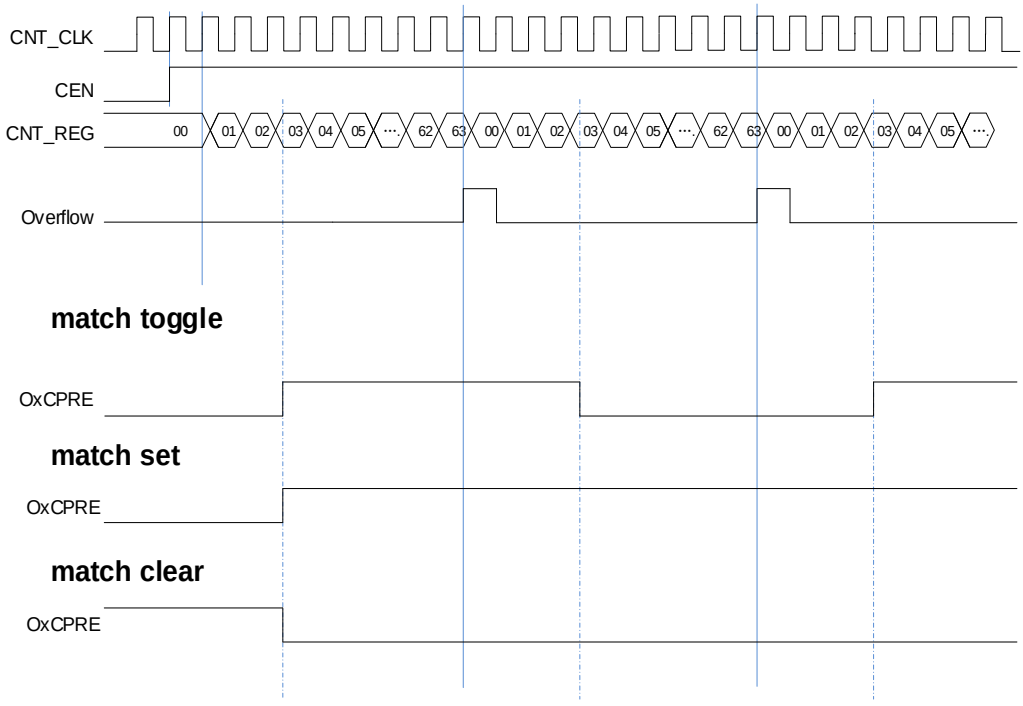
Step3: Interrupt/DMA-request enables configuration by CHxIE/CxCDE

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.

Figure 18-40. Output-compare in three modes show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 18-40. Output-compare in three modes



Output PWM function

In the output PWM function (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 18-41. Timing chart of EAPWM*** 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 18-42. Timing chart of CAPWM*** 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 18-41. Timing chart of EAPWM

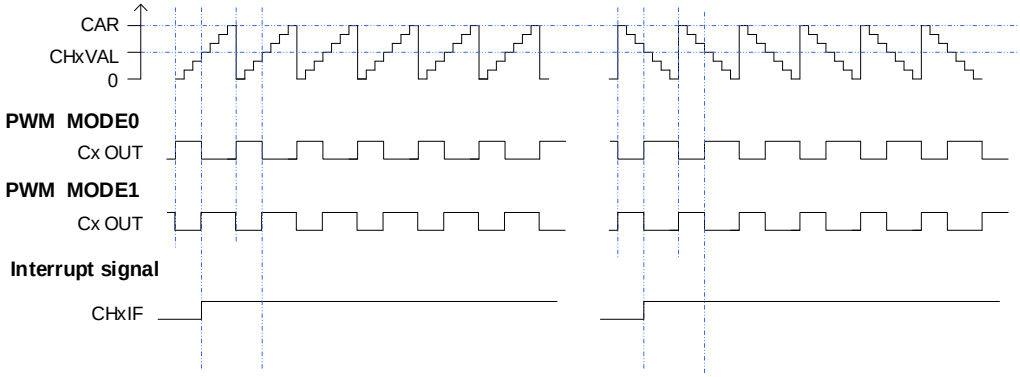
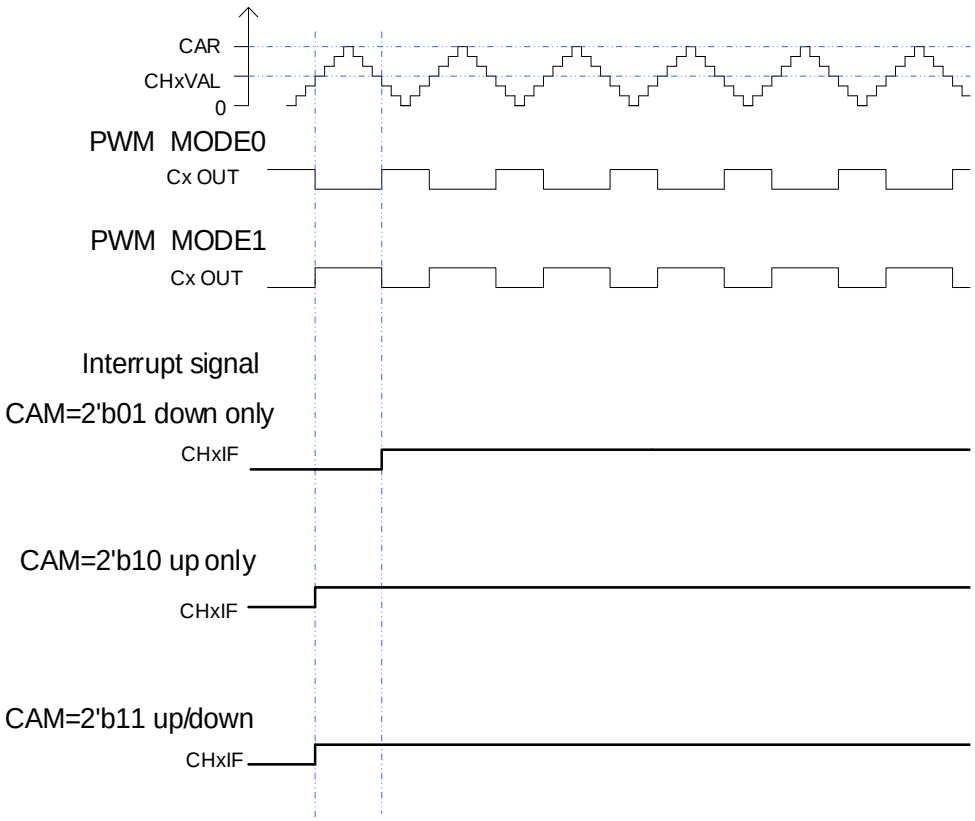


Figure 18-42. Timing chart of CAPWM



Channel output prepare signal

As is shown in ***Figure 18-39. Channel output compare principle (x=0,1,2,3)***, 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 ETIFE 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

Refer to ***Quadrature decoder***.

Hall sensor function

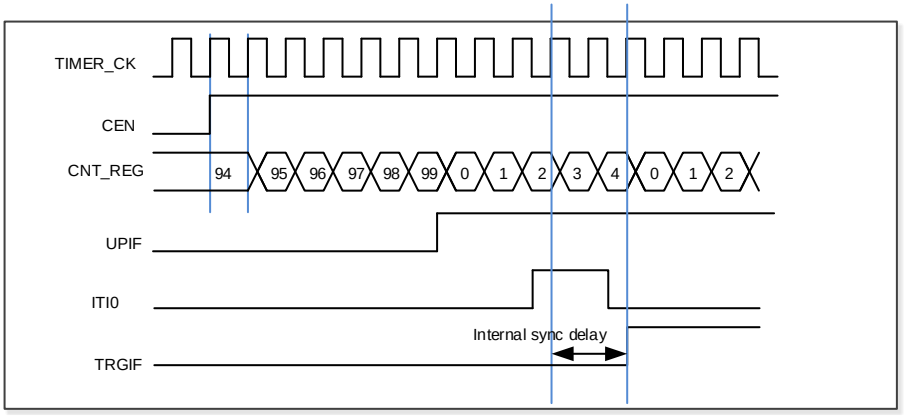
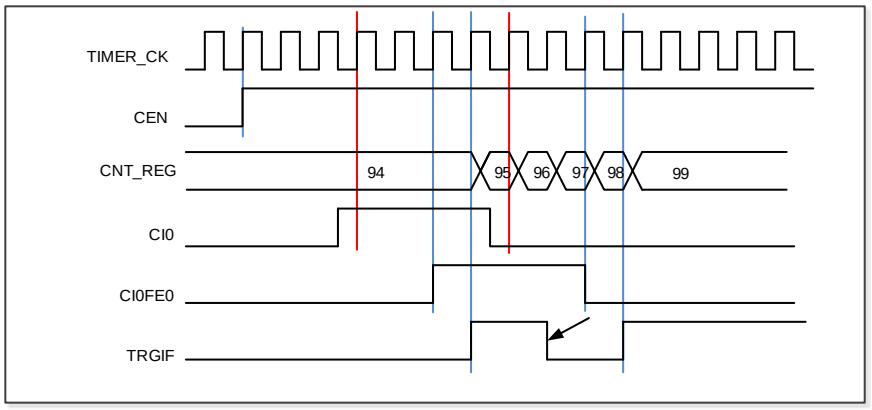
Refer to ***Hall sensor function***.

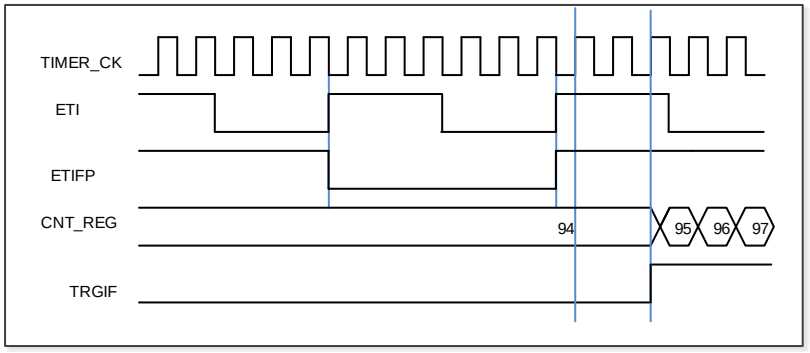
Master-slave management

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 18-5. Examples of slave mode

	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 CI0FE0 or CI1FE1 is selected as the trigger source, configure the CHxP and CHxNP for the polarity selection and inversion. If ETIFP is selected as the trigger source, configure the ETP for polarity selection and inversion.	For the ITIx, no filter and prescaler can be used. For the CIx, filter can be used by configuring CHxCAPFLT, no prescaler can be used. For the ETIFP, filter can be used by configuring ETFC and prescaler can be used by configuring ETPSC.
Exam1	**Restart mode** The counter will be cleared and restart when a rising edge of trigger input comes.	TRGS[2:0] = 3'b000 ITI0 is selected.	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 18-43. Restart mode
 
</td></tr>
<tr><td rowspan="2">Exam2</td><td>Pause mode
The counter will be paused when the trigger input is low, and it will start when the trigger input is high.</td><td>TRGS[2:0]=3’b101
CI0FE0 is selected.</td><td>TI0S=0 (Non-xor)
[CH0NP=0, CH0P=0]
CI0FE0 does not invert. The capture event will occur on the rising edge only.</td><td>Filter is bypassed in this example.</td></tr>
<tr><td colspan="4">Figure 18-44. Pause mode
 
</td></tr>
<tr><td>Exam3</td><td>Event mode
The counter will start to count when a rising edge of trigger input comes.</td><td>TRGS[2:0] =3’b111
ETIFP is selected.</td><td>ETP = 0, the polarity of ETI does not change.</td><td>ETPSC = 1, ETI is divided by 2.
ETFC = 0, ETI does not filter.</td></tr>
</table>

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	Figure 18-45. Event mode 			

Single pulse mode

Refer to ***Single pulse mode***.

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 oneword, onetime 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®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL0 register set to 1, the TIMERx counter stops.

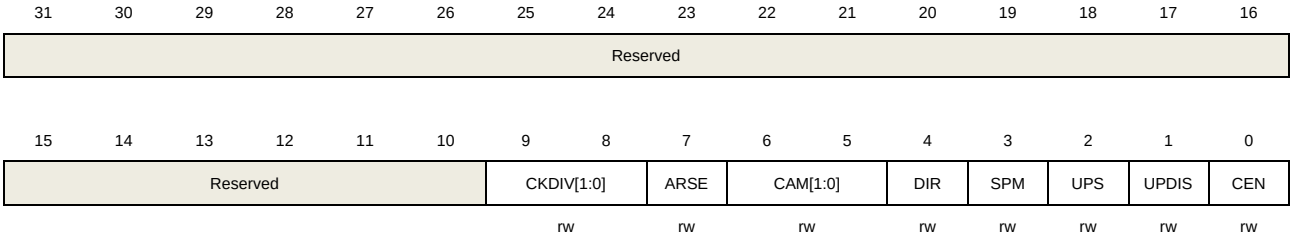
18.2.5. TIMERx registers(x=1, 2, 3, 4)

TIMER1 base address: 0x4000 0000
TIMER2 base address: 0x4000 0400
TIMER3 base address: 0x4000 0800
TIMER4 base address: 0x4000 0C00

Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

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



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 factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK\_TIMER}$ 01: $f_{DTS}= f_{CK\_TIMER}$ /2 10: $f_{DTS}= f_{CK\_TIMER}$ /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 counting down, CHxF bit 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 counting up, CHxF bit 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 counting up and counting down, CHxF bit 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 If the timer work in center-aligned mode or quadrature decode mode, this bit is read only.
3	SPM	Single pulse mode. 0: Single pulse mode disable. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: Update event disable. **Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.
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 quadrature decode mode.

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: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source: Master timer generate a reset the UPG bit in the TIMERx_SWEVG register is set 001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source : CEN control bit is set The trigger input in pause mode is high 010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit. 011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output. 100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE. 101: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O1CPRE. 110: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O2CPRE. 111: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O3CPRE.
3	DMAS	DMA request source selection 0: When capture or compare event occurs, the DMA request of channel x is sent 1: When update event occurs, the DMA request of channel x is sent.

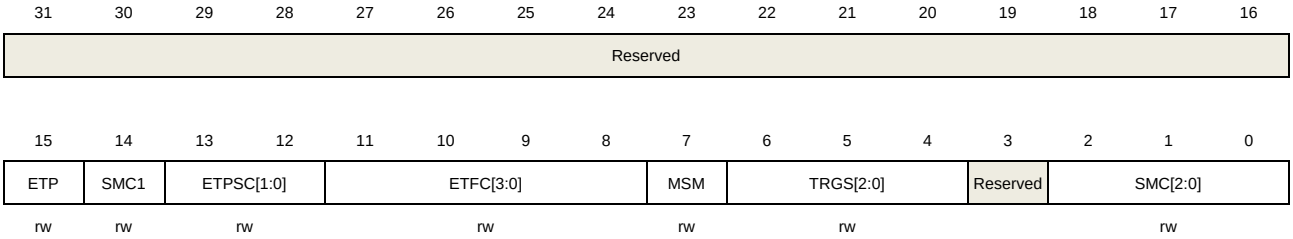
2:0	Reserved	Must be kept at reset value.

Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08

Reset value: 0x0000 0000

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



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 rising edge or high level . 1: ETI is active at falling edge or low level .
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. When the slave mode is configured as restart mode, pause mode or event mode, the timer can still work in the external clock 1 mode by setting this bit. But the TRGS bits must not be 3'b111 in this case. The clock source of the timer will be ETIFP if external clock mode 0 and external clock mode 1 are configured at the same time. **Note:** External clock mode 0 enable is in this register's SMC[2:0] bit-filed.
13:12	ETPSC[1:0]	The prescaler of external trigger 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: The prescaler is 2. 10: The prescaler is 4. 11: The prescaler is 8.
11:8	ETFC[3:0]	External trigger filter control

The external trigger can be filtered by digital filter and this bit-field configure the filtering capability.

Basic principle of digital filter: continuously sample the external trigger signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit-field, it is considered to be an effective level.

The filtering capability configuration is as follows:

EXTFC[3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{TIMER_CK}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS\_CK}/2$
4'b0101	8	
4'b0110	6	$f_{DTS\_CK}/4$
4'b0111	8	
4'b1000	6	$f_{DTS\_CK}/8$
4'b1001	8	
4'b1010	5	$f_{DTS\_CK}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS\_CK}/32$
4'b1110	6	
4'b1111	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: ITI0

001: ITI1

010: ITI2

011: ITI3

100: CI0F_ED

101: CI0FE0

110: CI1FE1

111: 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 CI0FE0 edge, while the direction depends on CI1FE1 level. 010: Quadrature decoder mode 1.The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 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 an update event is generated 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 clock when it is low. 110: Event mode. A rising edge of the trigger input enables the counter. 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	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	CH2IF	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 on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can 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. If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 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									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 CH0COMFEN 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 programmed as output mode 01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1 10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1 11: Channel 1 is programmed as input mode, IS1 is connected to ITS. **Note:** When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	Channel 0 output compare clear enable. When this bit is set, if the ETIFP signal is detected as high level, the O0CPRE signal will be cleared. 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 specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity 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 is equals to the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced to low level. 101: Force high. O0CPRE is forced to high level. 110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise. 111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise. If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.
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 verifying the shadow register only in single pulse mode (when SPM=1)
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. 1: Channel 0 output quickly compare enable.
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 programmed as output mode 01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0 10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0 11: Channel 0 is programmed as input mode, IS0 is connected to ITS **Note:** When CH0MS[1:0]=11, it is necessary to select an internal trigger input 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 2 input capture filter control The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI2 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH2CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}

4’b0010	4	
4’b0011	8	
4’b0100	6	$f_{DTS}/2$
4’b0101	8	
4’b0110	6	$f_{DTS}/4$
4’b0111	8	
4’b1000	6	$f_{DTS}/8$
4’b1001	8	
4’b1010	5	$f_{DTS}/16$
4’b1011	6	
4’b1100	8	
4’b1101	5	$f_{DTS}/32$
4’b1110	6	
4’b1111	8	

3:2 CH0CAPPSC[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, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on 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 CH0COMFEN 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 programmed as output mode 01: Channel 3 is programmed as input mode, IS3 is connected to CI3FE3 10: Channel 3 is programmed as input mode, IS3 is connected to CI2FE3 11: Channel 3 is programmed as input mode, IS3 is connected to ITS. **Note:** When CH3MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH2COMCEN	Channel 2 output compare clear enable. When this bit is set, if the ETIFP signal is detected as high level, the O2CPRE signal will be cleared. 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 specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP 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 is equals to the output compare register TIMERx_CH2CV. 010: Clear the channel output. O2CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH2CV. 011: Toggle on match. O2CPRE toggles when the counter is equals to the output compare register TIMERx_CH2CV. 100: Force low. O2CPRE is forced to low level. 101: Force high. O2CPRE is forced to high level. 110: PWM mode 0. When counting up, O2CPRE is high when the counter is smaller than TIMERx_CH2CV, and low otherwise. When counting down, O2CPRE is low when the counter is larger than TIMERx_CH2CV, and high otherwise. 111: PWM mode 1. When counting up, O2CPRE is low when the counter is smaller

		than TIMERx_CH2CV, and high otherwise. When counting down, O2CPRE is high when the counter is larger than TIMERx_CH2CV, and low otherwise. If configured in PWM mode, the O2CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.
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 verifying the shadow register only in single pulse mode (when SPM=1)
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. 1: Channel 2 output quickly compare enable.
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 programmed as output mode 01: Channel 2 is programmed as input mode, IS2 is connected to CI2FE2 10: Channel 2 is programmed as input mode, IS2 is connected to CI3FE2 11: Channel 2 is programmed as input mode, IS2 is connected to ITS. **Note:** When CH2MS[1:0]=11, it is necessary to select an internal trigger input 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

The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability.

Basic principle of digital filter: continuously sample the CI2 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.

The filtering capability configuration is as follows:

CH2CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	
4'b1111	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, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on 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
CH3NP	Reserved	CH3P	CH3EN	CH2NP	Reserved	CH2P	CH2EN	CH1NP	Reserved	CH1P	CH1EN	CH0NP	Reserved	CH0P	CH0EN
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	CH3NP	Channel 3 complementary output polarity Refer to CH0NP description
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	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	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
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	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit should be keep reset value. When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0.
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 high level is active level 1: Channel 0 low level is active level When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted. [CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not 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) (x=1)

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

Bits	Fields	Descriptions
31:0	CNT[31:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

Counter register (TIMERx_CNT) (x=2,3,4)

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 TIMER_CK 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) (x=1)

Address offset: 0x2C
Reset value: 0x0000 0000 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
CARL[31:16]															
rw															

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

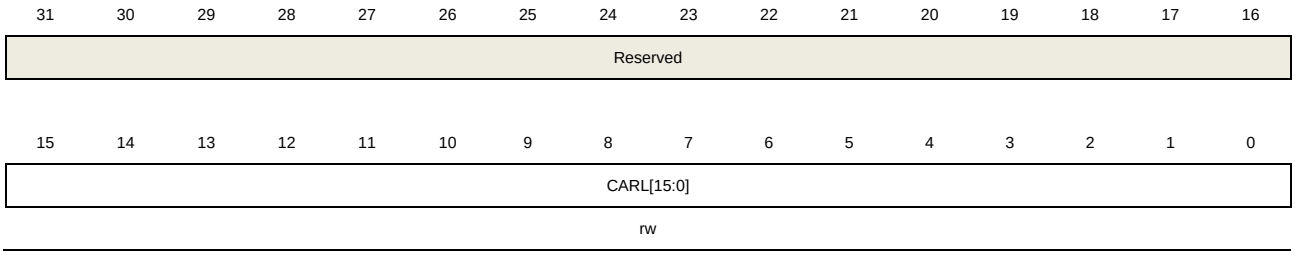
rw

Bits	Fields	Descriptions
31:0	CARL[31:0]	Counter auto reload value This bit-filed specifies the auto reload value of the counter.

Counter auto reload register (TIMERx_CAR) (x=2,3,4)

Address offset: 0x2C
Reset value: 0x0000 0000

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

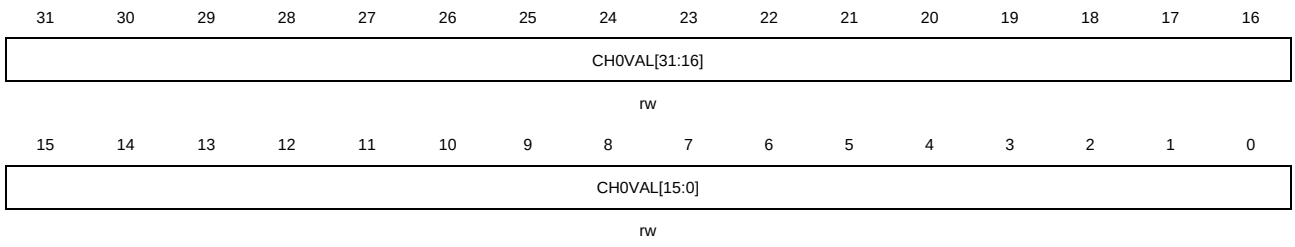


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CARL[15:0]	Counter auto reload value This bit-filed specifies the auto reload value of the counter.

Channel 0 capture/compare value register (TIMERx_CH0CV) (x=1)

Address offset: 0x34
Reset value: 0x0000 0000 0000

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



Bits	Fields	Descriptions
31:0	CH0VAL[31:0]	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 0 capture/compare value register (TIMERx_CH0CV) (x=2,3,4)

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) (x=1)

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
CH1VAL[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:0	CH1VAL[31:0]	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 1 capture/compare value register (TIMERx_CH1CV) (x=2,3,4)

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) (x=1)

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
CH2VAL[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH2VAL[15:0]															
rw															

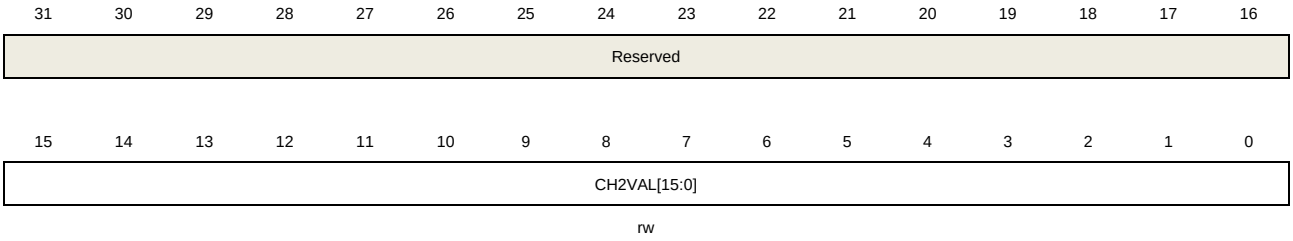
Bits	Fields	Descriptions
31:0	CH2VAL[31:0]	Capture or compare value of channel 2 When channel 2 is configured in input mode, this bit-filed indicates the counter value

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 2 capture/compare value register (TIMERx_CH2CV) (x=2,3,4)

Address offset: 0x3C
Reset value: 0x0000 0000

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

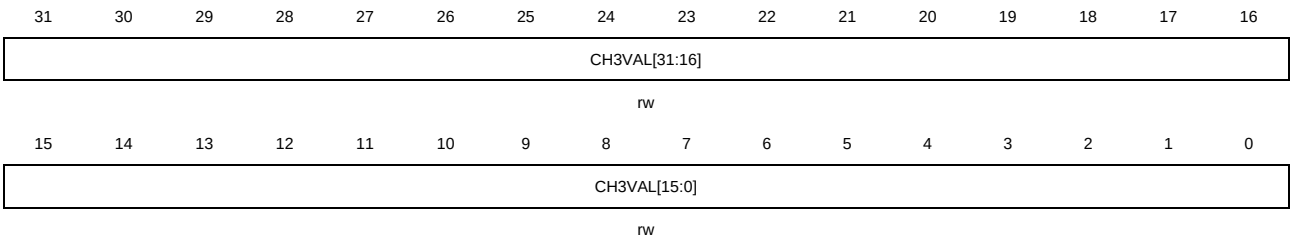


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CH2VAL[15:0]	Capture or compare value of channel 2 When channel 2 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 2 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) (x=1)

Address offset: 0x40
Reset value: 0x0000 0000

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



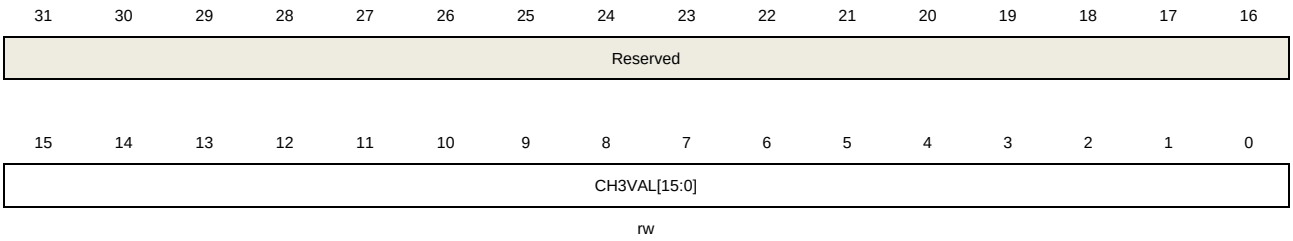
Bits	Fields	Descriptions
31:0	CH3VAL[31:0]	Capture or compare value of channel 3

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.

Channel 3 capture/compare value register (TIMERx_CH3CV) (x=2,3,4)

Address offset: 0x40
Reset value: 0x0000 0000

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

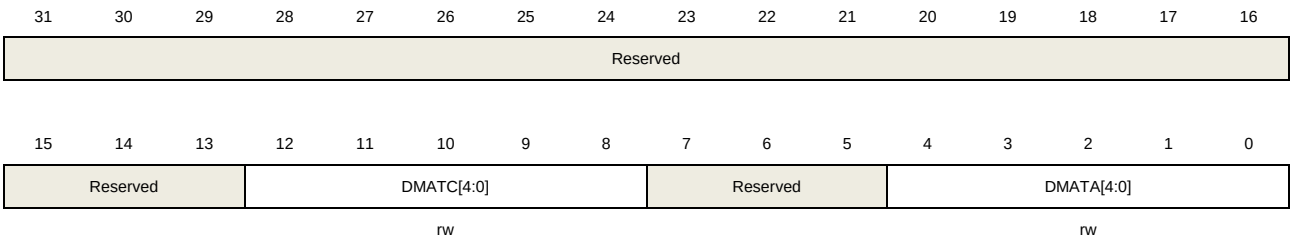


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CH3VAL[15:0]	Capture or compare value of channel 3 When channel3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 3 is 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).



Bits	Fields	Descriptions

31:13	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This filed defines the number(n) of the register that DMA will access(R/W), n = (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001.
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.

DMA transfer buffer register (TIMERx_DMATB)(x=1)

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

Bits	Fields	Descriptions
31:0	DMATB[31:0]	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.

DMA transfer buffer register (TIMERx_DMATB)(x=2,3,4)

Address offset: 0x4C
Reset value: 0x0000 0000

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

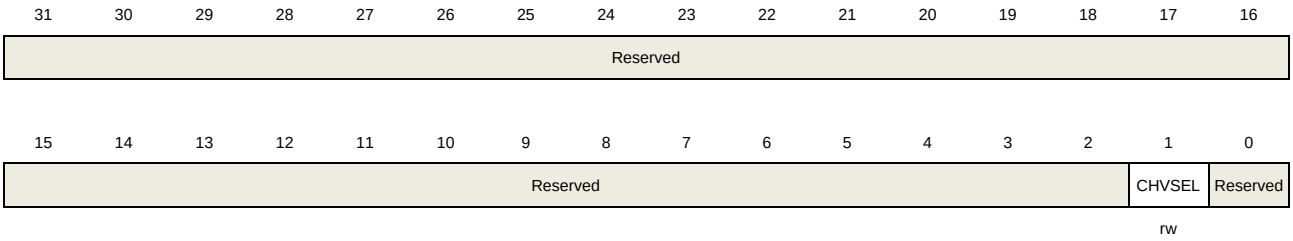
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMATB[15:0]															
rw															

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.

Configuration register (TIMERx_CFG)

Address offset: 0xFC
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CHVSEL	Write CHxVAL register selection This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	Reserved	Must be kept at reset value.

18.3. General level1 timer (TIMERx, x=8, 11)

18.3.1. 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.

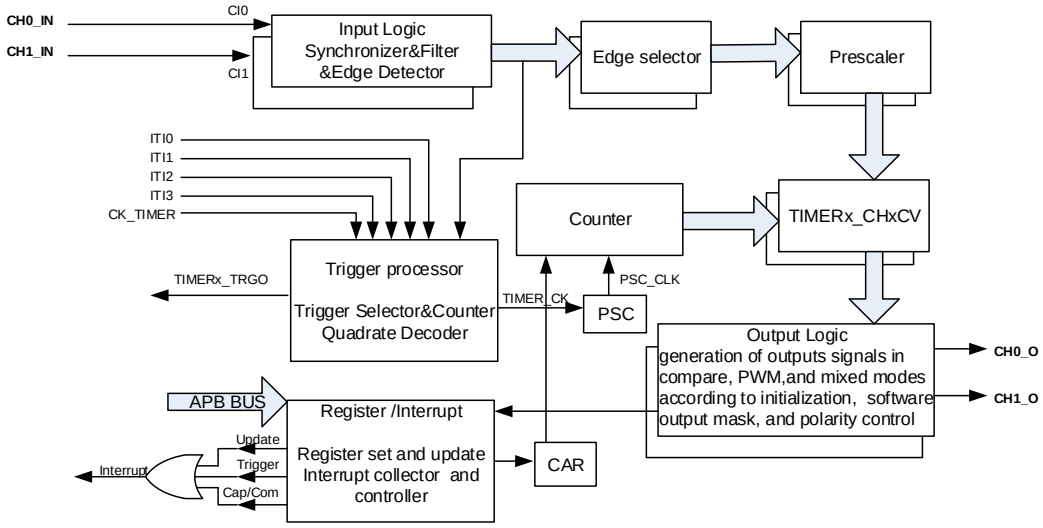
18.3.2. Characteristics

- Total channel num: 2.
- Counter width: 16 bit.
- Source of count clock is selectable:
  internal clock, internal trigger, external input.
- counter mode: count up only.
- 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 management.

18.3.3. Block diagram

Figure 18-46. General level1 timer block diagram provides details on the internal configuration of the general level1 timer.

Figure 18-46. General level1 timer block diagram



18.3.4. Function overview

Clock source configuration

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]).

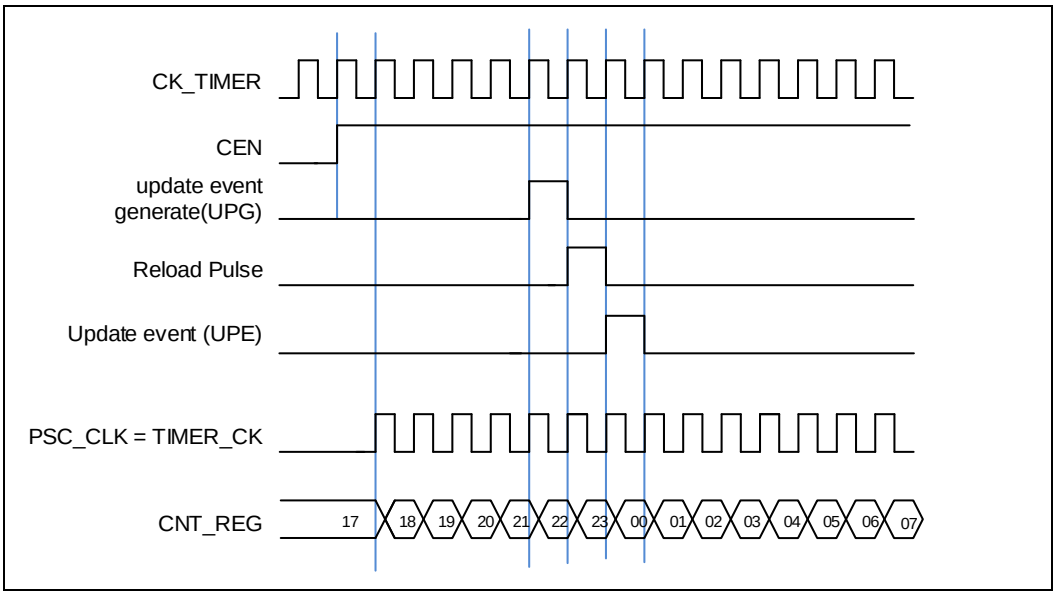
- SMC [2:0] == 3'b000. Internal timer clock CK_TIMER which is from module RCU.

The default internal clock source is the CK_TIMER used to drive the counter prescaler when the 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 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 SMC[2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK_TIMER is the counter prescaler driving clock source.

Figure 18-47. Timing chart of internal clock divided by 1



- SMC [2:0] == 3'b111 (external clock mode 0). External input pin source

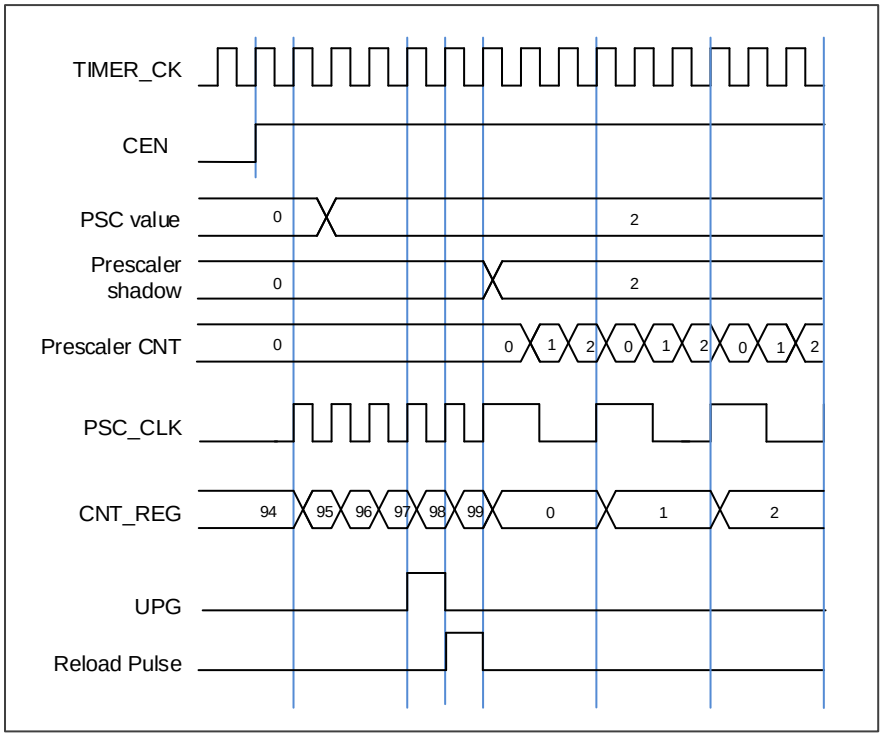
The TIMER_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CI0/TIMERx_CI1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

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.

Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 18-48. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again. 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 the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

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

Figure 18-49. Timing chart of up counting mode, PSC=0/2 and ***Figure 18-50. Timing chart of up counting mode, change TIMERx_CAR ongoing*** show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

Figure 18-49. Timing chart of up counting mode, PSC=0/2

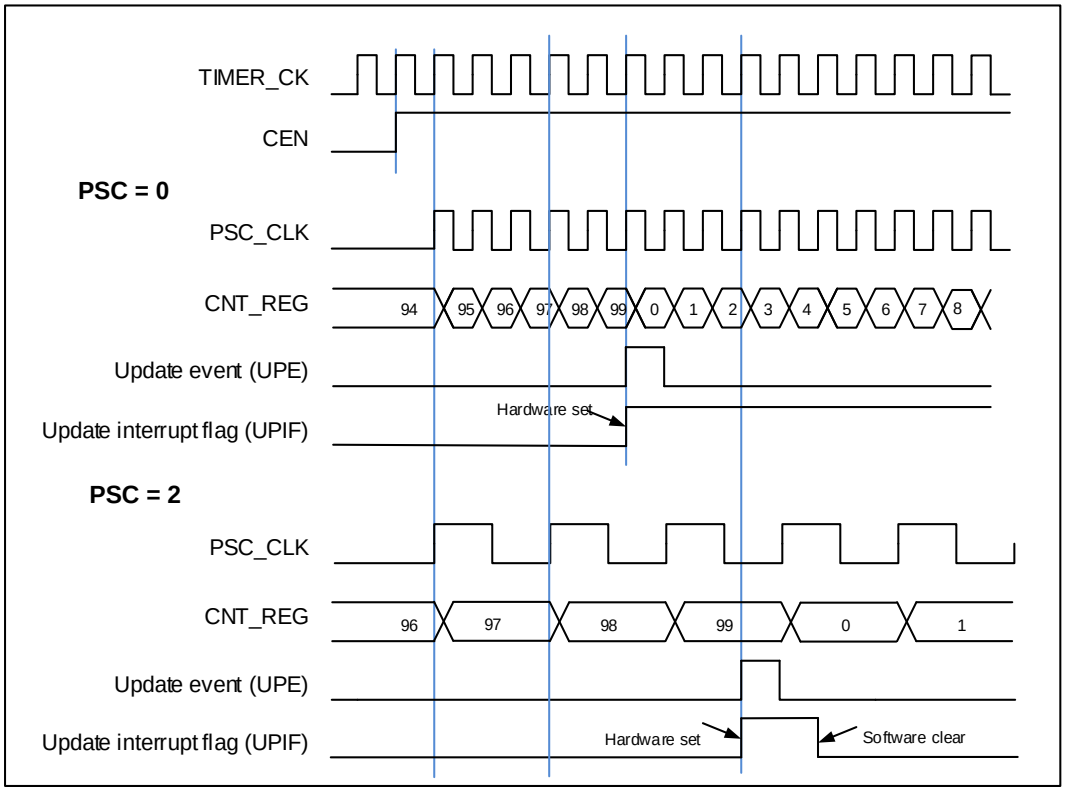
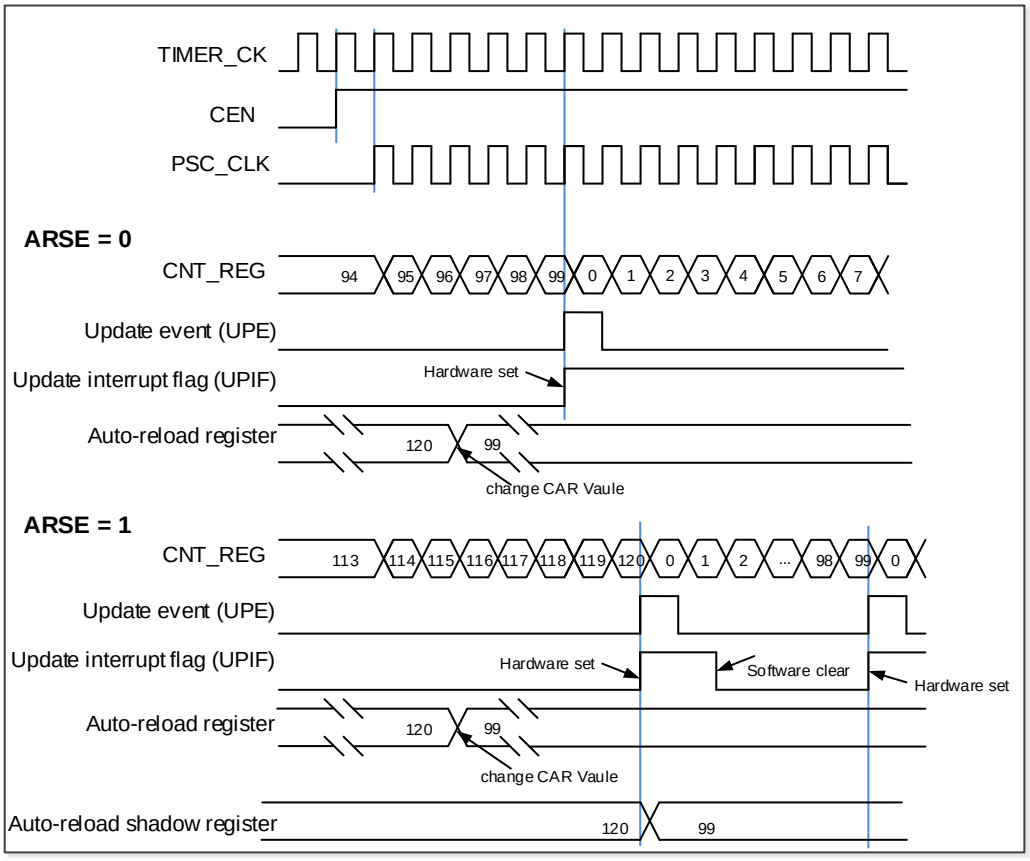


Figure 18-50. Timing chart of up counting mode, change TIMERx_CAR ongoing



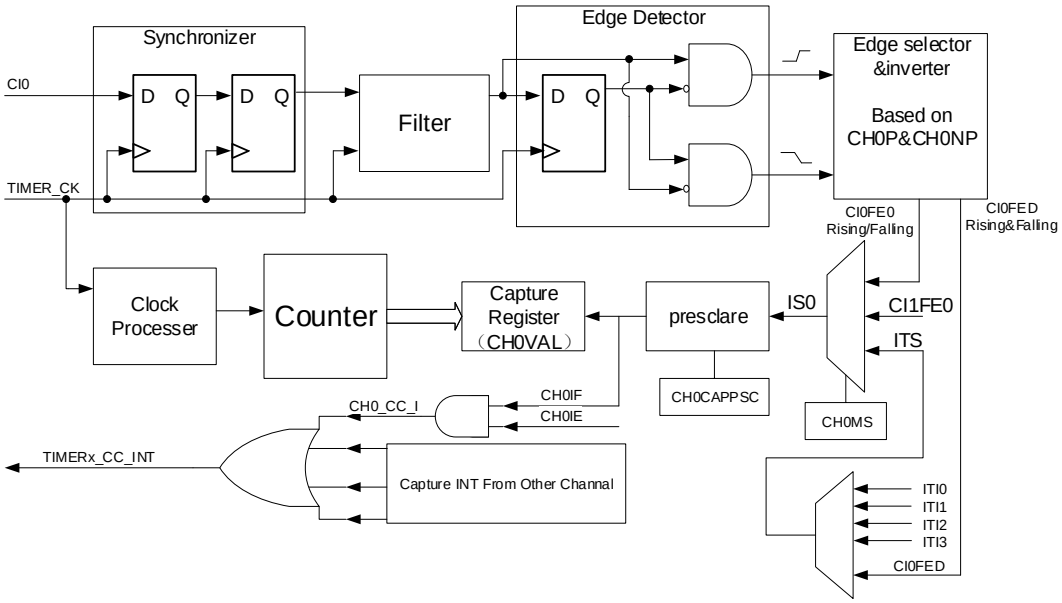
Input capture and output compare channels

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.

- Channel Input capture function

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 18-51. Channel input capture principle



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, TIMERx_CHxCV 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 channel input capture function 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.

- **Channel output compare function**

Figure 18-52. Channel output compare principle (x=0,1)

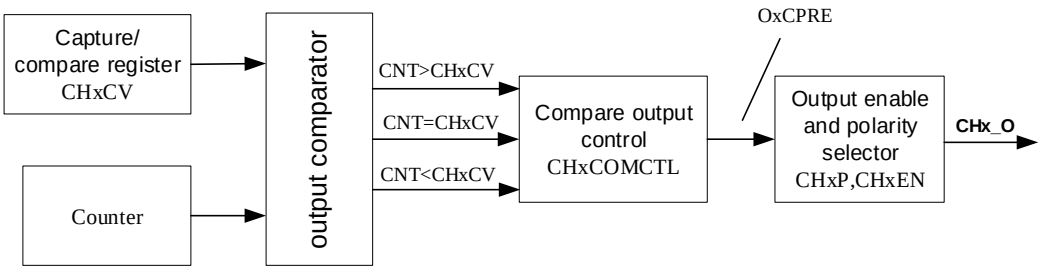


Figure 18-52. Channel output compare principle (x=0,1) shows the principle circuit of channels output compare function. The relationship between the channel output signal CHx_O and the OxCPRE signal (more details refer to *Channel output prepare signal*) is described as blew: The active level of O0CPRE is high, the output level of CH0_O depends on OxCPRE signal, CHxP bit and CH0P bit (please refer to theTIMERx_CHCTL2 register for more details).For example, configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxEN=1 (the output of CHx_O is enabled),

If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;
If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

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

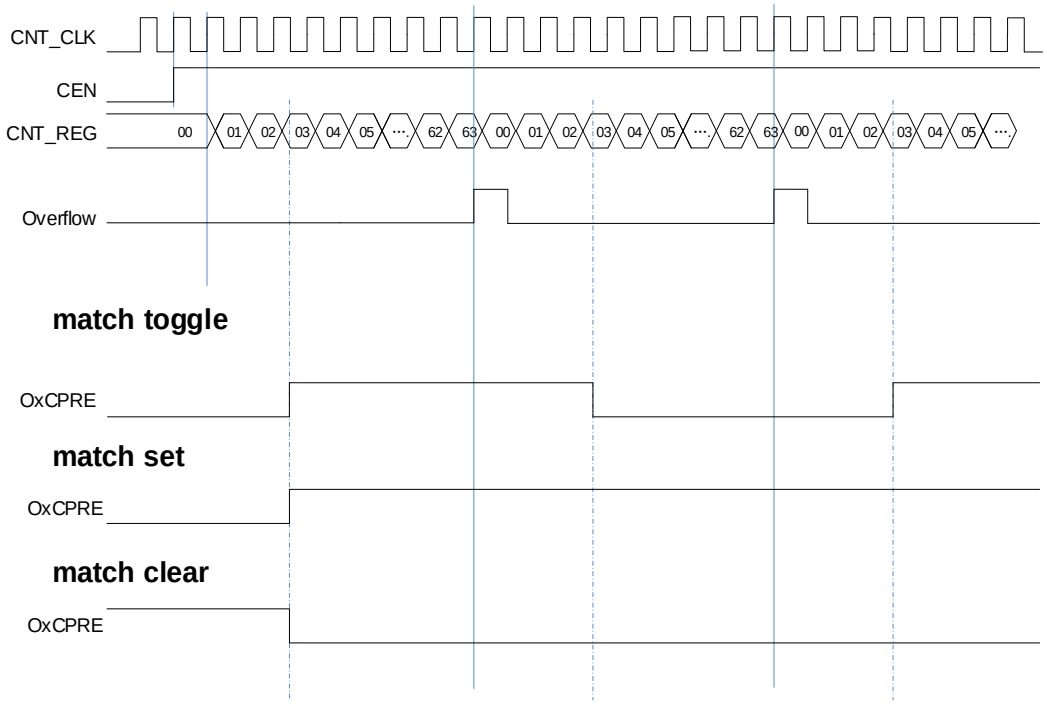
Step3: Interrupt/DMA-request enables configuration by CHxIE/CxCDE

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.

Figure 18-53. Output-compare under three modes show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 18-53. Output-compare under three modes



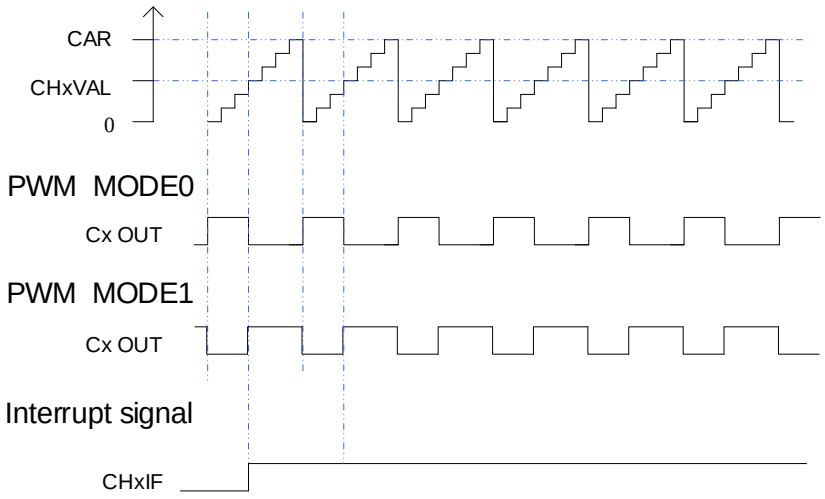
Output PWM function

In the output PWM function (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.

The period is determined by TIMERx_CAR and duty cycle is determined by TIMERx_CHxCV. ***Figure 18-54. PWM mode timechart*** shows the PWM output mode 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 18-54. PWM mode timechart



Channel output prepare signal

As is shown in ***Figure 18-52. Channel output compare principle (x=0,1)***, 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.

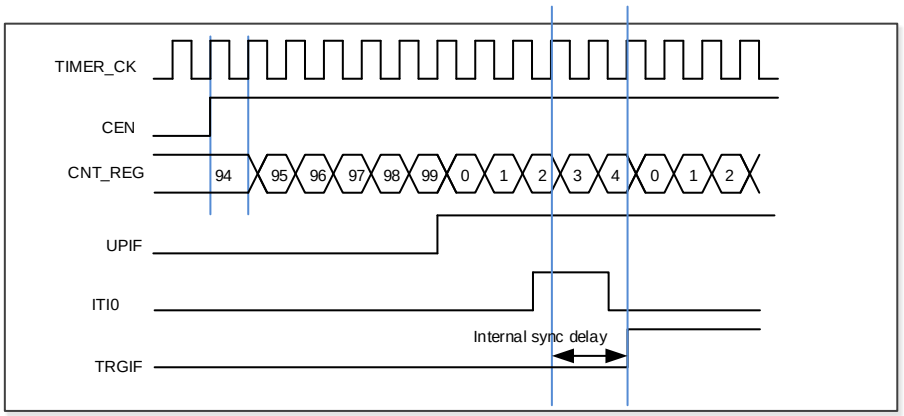
The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx_CHxCV content. 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.

Master-slave management

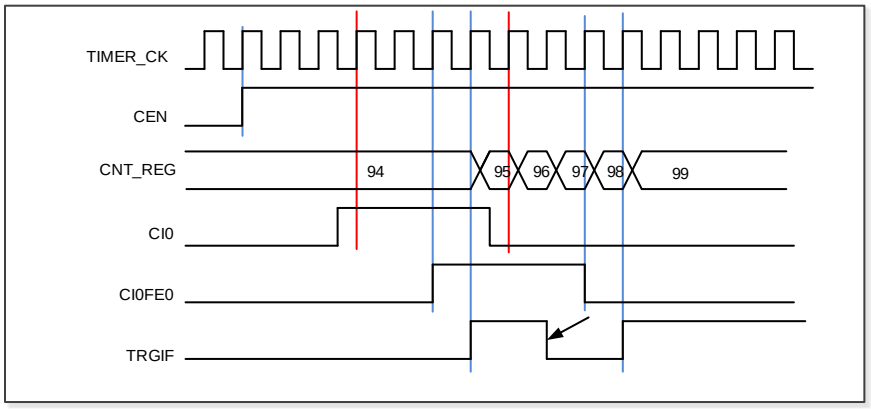
The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMC [2:0] in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx_SMCFG register.

Table 18-6. Examples of slave mode

	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: Reserved	If CI0FE0 or CI1FE1 is selected as the trigger source, configure the CHxP and CHxNP for the polarity selection and inversion.	For the ITIx, no filter and prescaler can be used. For the CIx, filter can be used by configuring CHxCAPFLT, no prescaler can be used.
Exam1	**Restart mode** The counter will be cleared and restart when a rising edge of trigger input comes.	TRGS[2:0] = 3'b000 ITI0 is selected.	For ITI0, no polarity selector can be used.	For the ITI0, no filter and prescaler can be used.
	Figure 18-55. Restart mode 			
Exam2	**Pause mode** The counter will be paused when the trigger input is low,	TRGS[2:0]=3'b101 CI0FE0 is selected.	CH0P=0, CI0FE0 does not invert. The capture event will occur on the	Filter is bypassed in this example.

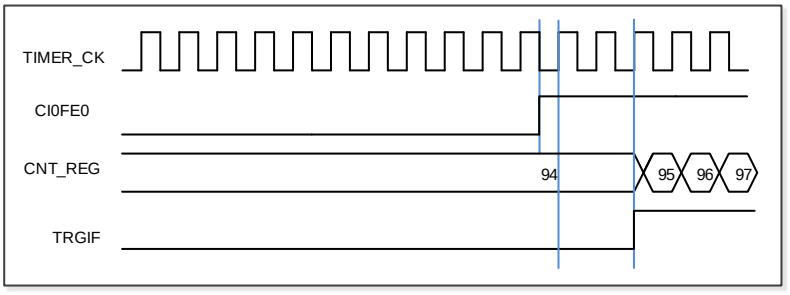
	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	and it will start when the trigger input is high.		rising edge only.	

Figure 18-56. Pause mode



Exam3	**Event mode** The counter will start to count when a rising edge of trigger input comes.	TRGS[2:0]=3'b101 CI0FE0 is selected.	CH0P=0, CI0FE0 does not invert. The capture event will occur on the rising edge only.	Filter is bypassed in this example.

Figure 18-57. Event mode



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. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

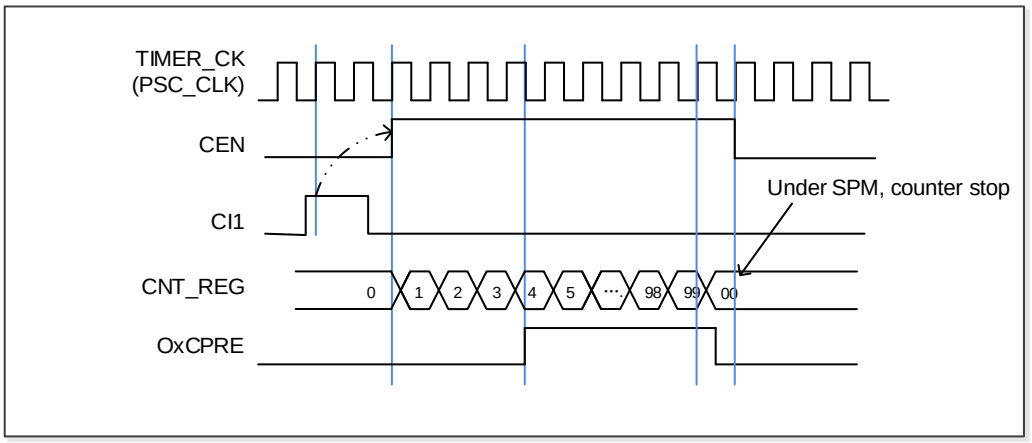
Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1 using software. Setting the 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.

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 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 18-58. Single pulse mode TIMERx_CHxCV = 4 TIMERx_CAR=99 shows an example.

Figure 18-58. Single pulse mode TIMERx_CHxCV = 4 TIMERx_CAR=99



Timers interconnection

Refer to *Advanced timer (TIMERx, x=0, 7)*.

Timer debug mode

When the Cortex®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL0 register set to 1, the TIMERx counter stops.

18.3.5. TIMERx registers(x=8, 11)

TIMER8 base address: 0x4001 4C00

TIMER11 base address: 0x4000 1800

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	Reserved			SPM	UPS	UPDIS	CEN
						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 factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK\_TIMER}$ 01: $f_{DTS}= f_{CK\_TIMER}$ /2 10: $f_{DTS}= f_{CK\_TIMER}$ /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
6:4	Reserved	Must be kept at reset value.
3	SPM	Single pulse mode. 0: Single pulse mode disable. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests:

		The counter generates an overflow or underflow event
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: Update event disable. **Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.
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 quadrature decode mode.

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	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: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: Reserved. 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: Reserved. 010: Reserved. 011: Reserved. 100: Restart mode. The counter is reinitialized and an update event is generated 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 clock when it is low. 110: Event mode. A rising edge of the trigger input enables the counter. 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 on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on

		trigger input can 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. If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 0: No Channel 0interrupt 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									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
Reserved	CH1COMCTL[2:0]			CH1COMSEN	CH1COMFEN	CH1MS[1:0]		Reserved	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
31:15	Reserved	Must be kept at reset value.
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 CH0COMFEN 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 programmed as output mode 01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1 10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1 11: Channel 1 is programmed as input mode, IS1 is connected to ITS. **Note:** When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	Reserved	Must be kept at reset value.
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity 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 is equals to the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced to low level. 101: Force high. O0CPRE is forced to high level. 110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise. 111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise. If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.
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 verifying the shadow register only in single pulse mode (when SPM=1)
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. 1: Channel 0 output quickly compare enable.
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 programmed as output mode 01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0 10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0 11: Channel 0 is programmed as input mode, IS0 is connected to ITS **Note:** When CH0MS[1:0]=11, it is necessary to select an internal trigger input 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 The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI0 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH0CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	

		4’b0110	6	f_{DTS}/4
		4’b0111	8	
		4’b1000	6	f_{DTS}/8
		4’b1001	8	
		4’b1010	5	f_{DTS}/16
		4’b1011	6	
		4’b1100	8	
		4’b1101	5	f_{DTS}/32
		4’b1110	6	
		4’b1111	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, input capture occurs on every channel input edge 01: The input capture occurs on every 2 channel input edges 10: The input capture occurs on every 4 channel input edges 11: The input capture occurs on 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								CH1NP	Reserved	CH1P	CH1EN	CH0NP	Reserved	CH0P	CH0EN
								rw		rw	rw	rw		rw	rw

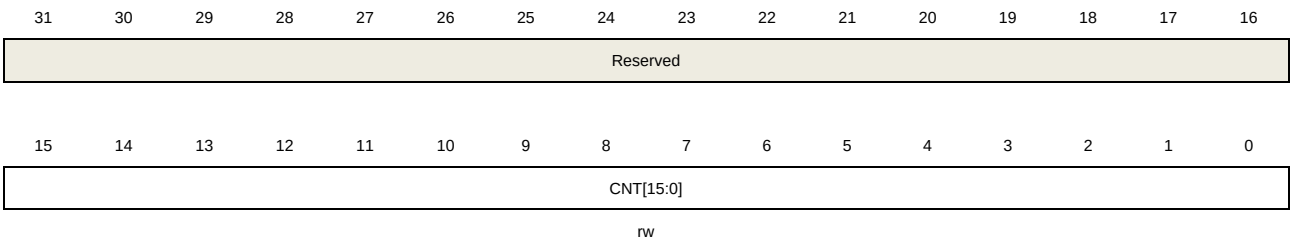
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
6	Reserved	Must be kept at reset value.
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description

4	CH1EN	Channel 1 capture/compare function enable Refer to CH1EN description
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit should be keep reset value. When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0.
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 high level is active level 1: Channel 0 low level is active level When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted. [CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not 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).

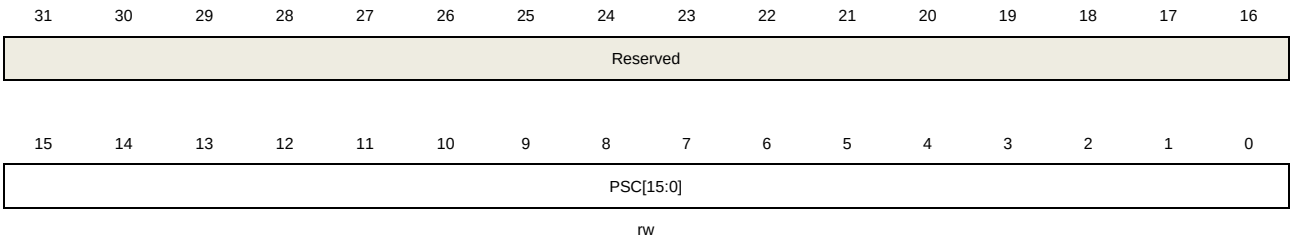


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

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

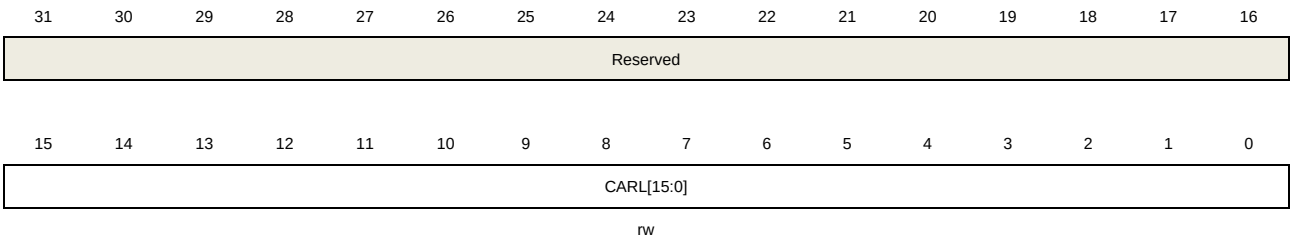


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	PSC[15:0]	Prescaler value of the counter clock The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.

Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0x0000 0000

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



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.

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															

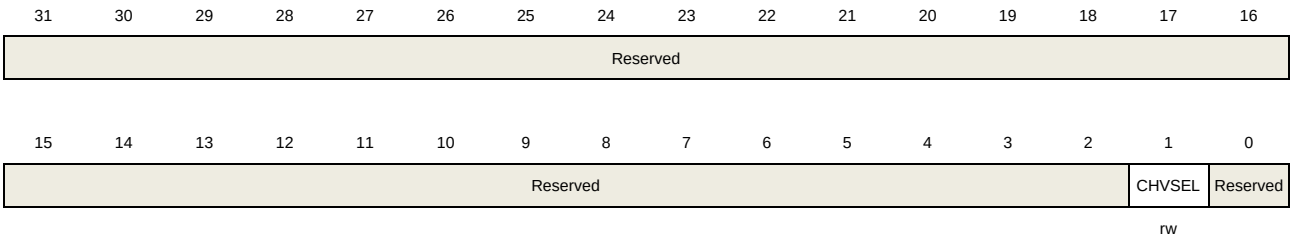
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CH1VAL[15:0]	Capture or compare value of channel1 When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.

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.

Configuration register (TIMERx_CFG)

Address offset: 0xFC
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CHVSEL	Write CHxVAL register selection This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	Reserved	Must be kept at reset value.

18.4. General level2 timer (TIMERx, x=9, 10, 12, 13)

18.4.1. 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.

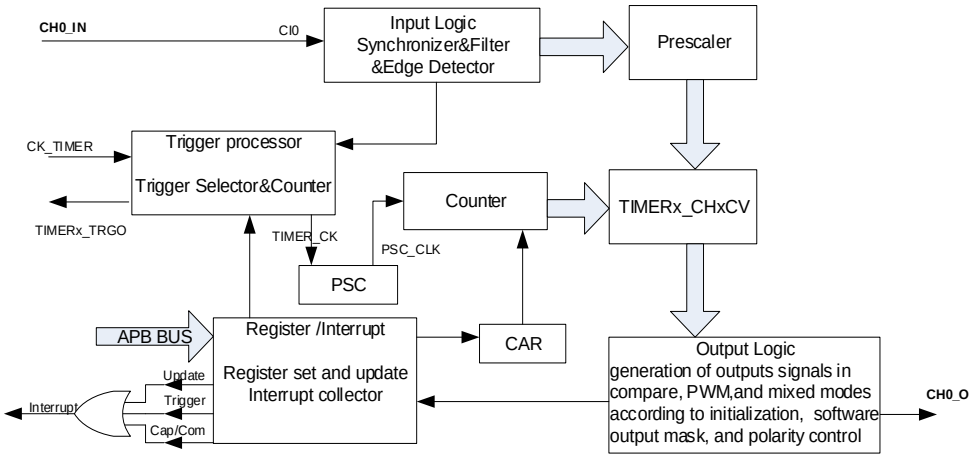
18.4.2. Characteristics

- Total channel num: 1.
- Counter width: 16bit.
- Source of count clock: internal clock.
- Counter mode: count up only.
- Programmable prescaler: 16 bits. Factor can be changed on the go.
- Each channel is user-configurable:
  Input capture mode, output compare mode, programmable and PWM mode.
- Auto-reload function.
- Interrupt output on: update event and compare/capture event.

18.4.3. Block diagram

Figure 18-59. General level2 timer block diagram provides details on the internal configuration of the general level2 timer.

Figure 18-59. General level2 timer block diagram



18.4.4. Function overview

Clock source configuration

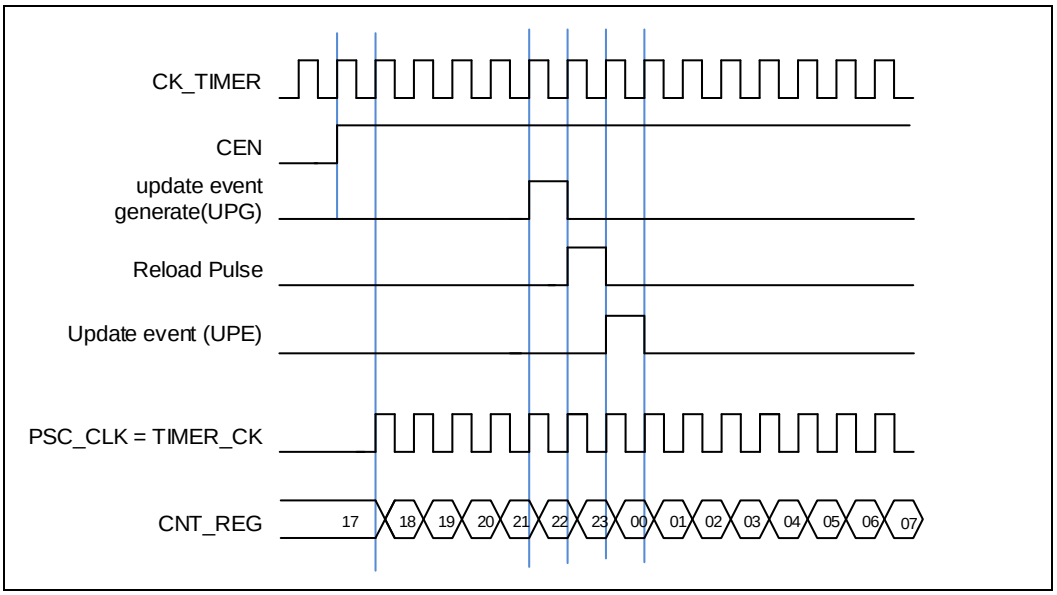
The general level2 TIMER can only being clocked by the CK_TIMER.

- 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

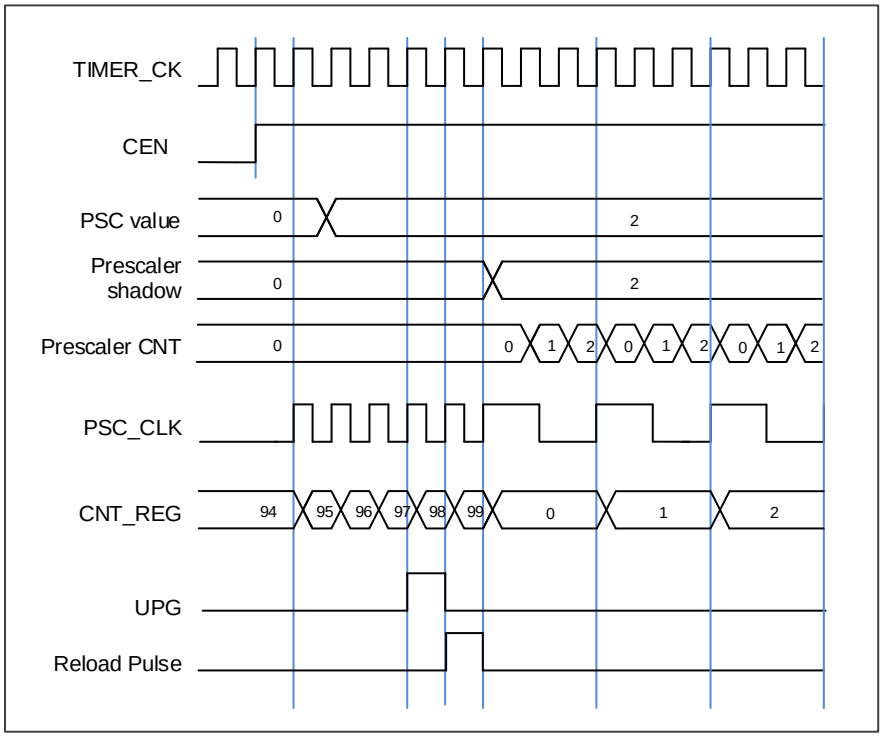
Figure 18-60. Timing chart of internal clock divided by 1



Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 18-61. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again. 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 the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

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

Figure 18-62. Timing chart of up counting mode, PSC=0/2 and *Figure 18-63. Timing chart of up counting mode, change TIMERx_CAR ongoing* show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

Figure 18-62. Timing chart of up counting mode, PSC=0/2

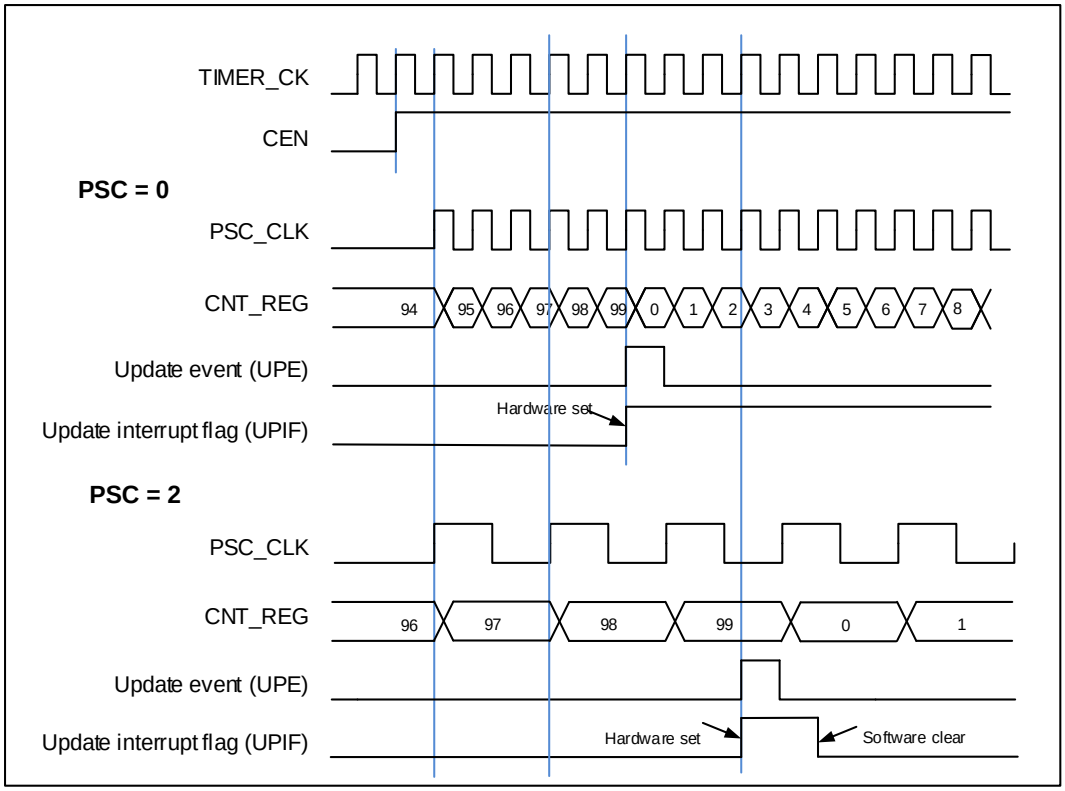
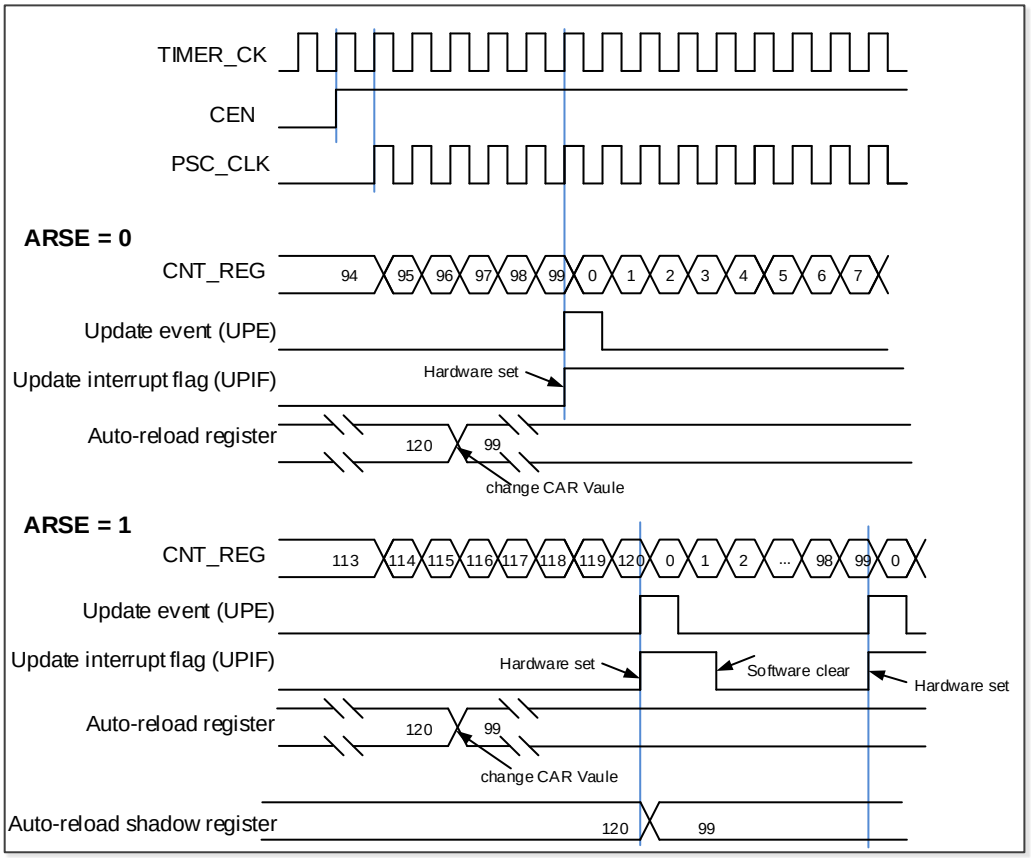


Figure 18-63. Timing chart of up counting mode, change TIMERx_CAR ongoing



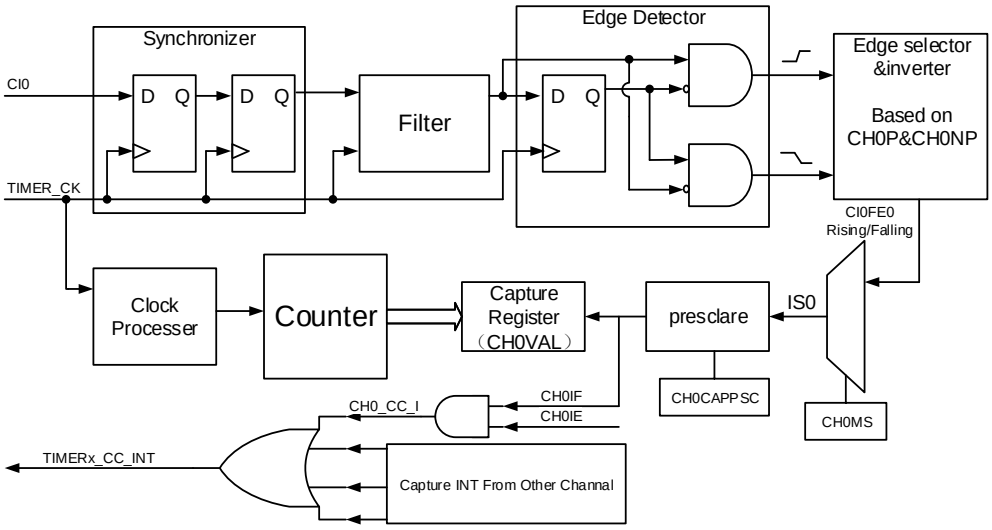
Input capture and output compare channels

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.

- Channel input capture function

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 18-64. Channel input capture principle



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, TIMERx_CHxCV 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 channel input capture function 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.

- **Channel output compare function**

Figure 18-65. Channel output compare principle

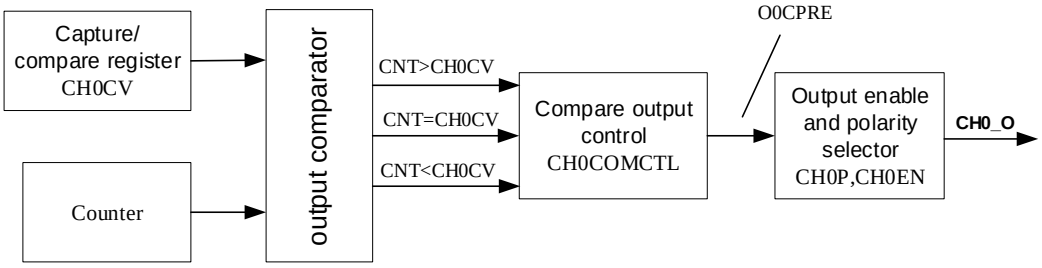


Figure 18-65. Channel output compare shows the principle circuit of channels output compare function. The relationship between the channel output signal CHx_O and the OxCPRE signal (more details refer to *Channel output prepare signal*) is described as blew: The active level of O0CPRE is high, the output level of CH0_O depends on OxCPRE signal, CHxP bit and CHxEN bit (please refer to the TIMERx_CHCTL2 register for more details).For example, configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxEN=1 (the output of CHx_O is enabled),

If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;
If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

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

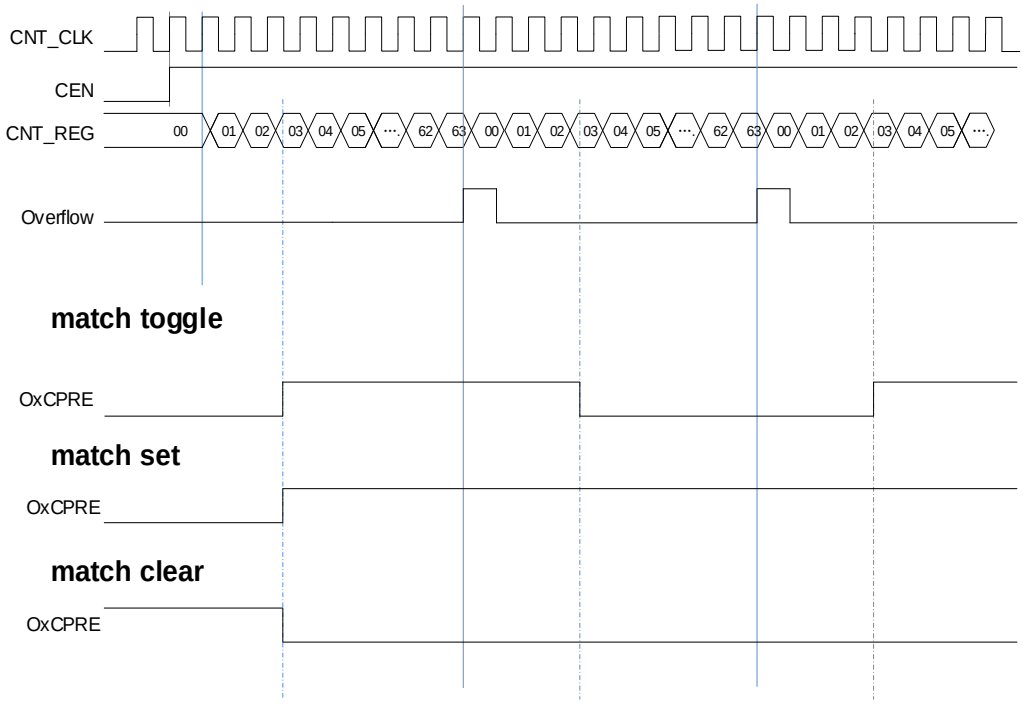
Step3: Interrupt/DMA-request enables configuration by CHxIE

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.

Figure 18-66. Output-compare under three modes show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 18-66. Output-compare under three modes



Channel output prepare signal

As is shown in *Figure 18-65. Channel output compare* , 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.

Timer debug mode

When the Cortex®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL0 register set to 1, the TIMERx counter stops.

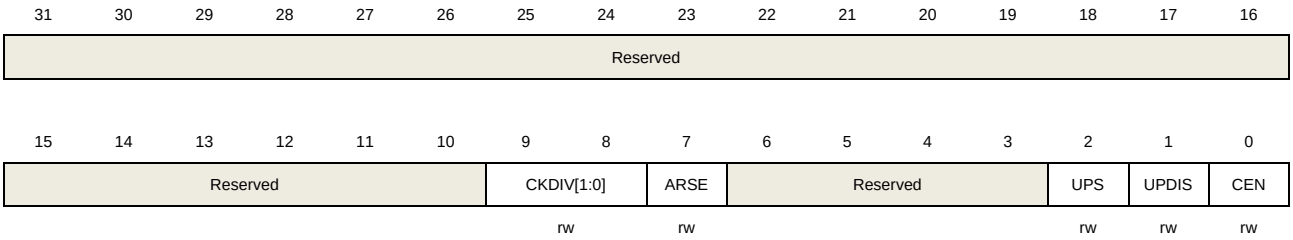
18.4.5. TIMERx registers(x=9, 10, 12, 13)

TIMER9 base address: 0x4001 5000
TIMER10 base address: 0x4001 5400
TIMER12 base address: 0x4000 1C00
TIMER13 base address: 0x4000 2000

Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

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



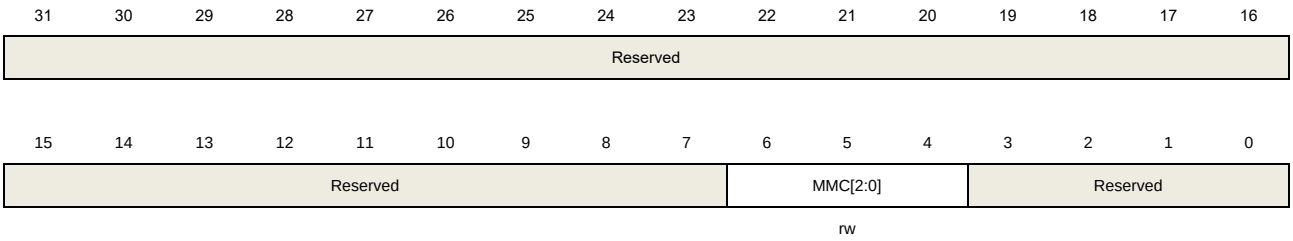
Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9:8	CKDIV[1:0]	Clock division The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK\_TIMER}$ 01: $f_{DTS}= f_{CK\_TIMER}$ /2 10: $f_{DTS}= f_{CK\_TIMER}$ /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
6:3	Reserved	Must be kept at reset value.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event

1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: Update event disable. **Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.
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 quadrature decode mode.

Control register 1 (TIMERx_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

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



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: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source: Master timer generate a reset the UPG bit in the TIMERx_SWEVG register is set 001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source : CEN control bit is set

		The trigger input in pause mode is high 010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit. 011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output. 100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE. 101: Reserved 110: Reserved 111: Reserved
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

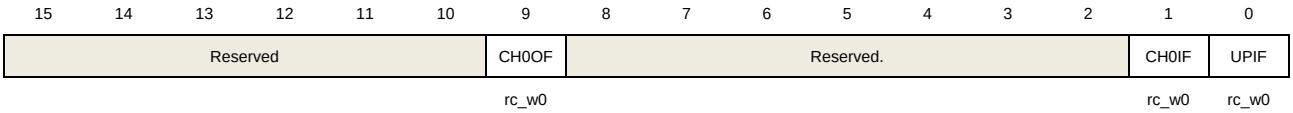
Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

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															

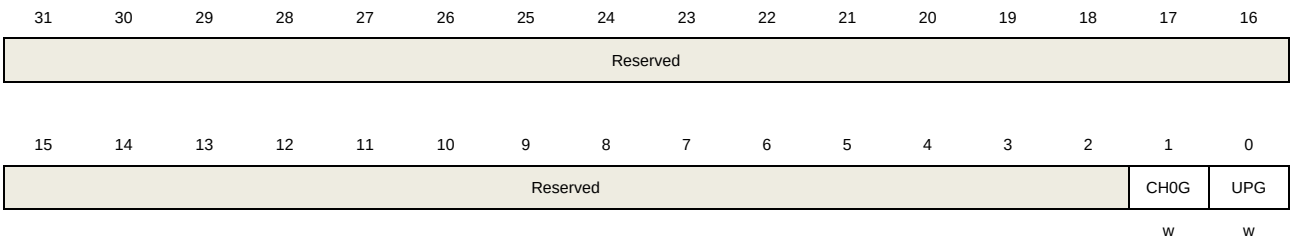


Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt 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. If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 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).



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 specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity 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 is equals to the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter is equals to the output

		compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced to low level. 101: Force high. O0CPRE is forced to high level. 110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise. 111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise. If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.
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 verifying the shadow register only in single pulse mode (when SPM=1)
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. 1: Channel 0 output quickly compare enable.
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: Reserved 11: Reserved

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 The CI0 input signal can be filtered by digital filter and this bit-field configure the

filtering capability.

Basic principle of digital filter: continuously sample the CI0 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.

The filtering capability configuration is as follows:

CH0CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	
4'b1111	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, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on 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..	CH0NP	Reserved	CH0P	CH0EN
	rw		rw	rw

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.
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 complementary output high level is active level 1: Channel 0 complementary output low level is active level When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0.
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 high level is active level 1: Channel 0 low level is active level When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted. [CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not 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 TIMER_CK 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.

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.

Configuration register (TIMERx_CFG)

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
Reserved															

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

rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CHVSEL	Write CHxVAL register selection This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	Reserved	Must be kept at reset value.

18.5. Basic timer (TIMERx, x=5, 6)

18.5.1. 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.

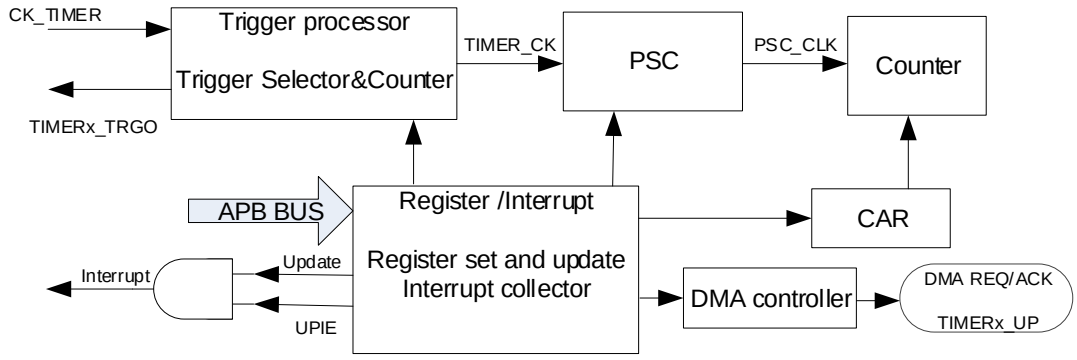
18.5.2. Characteristics

- 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.
- Auto-reload function.
- Interrupt output or DMA request on update event.

18.5.3. Block diagram

Figure 18-67. Basic timer block diagram provides details on the internal configuration of the basic timer.

Figure 18-67. Basic timer block diagram



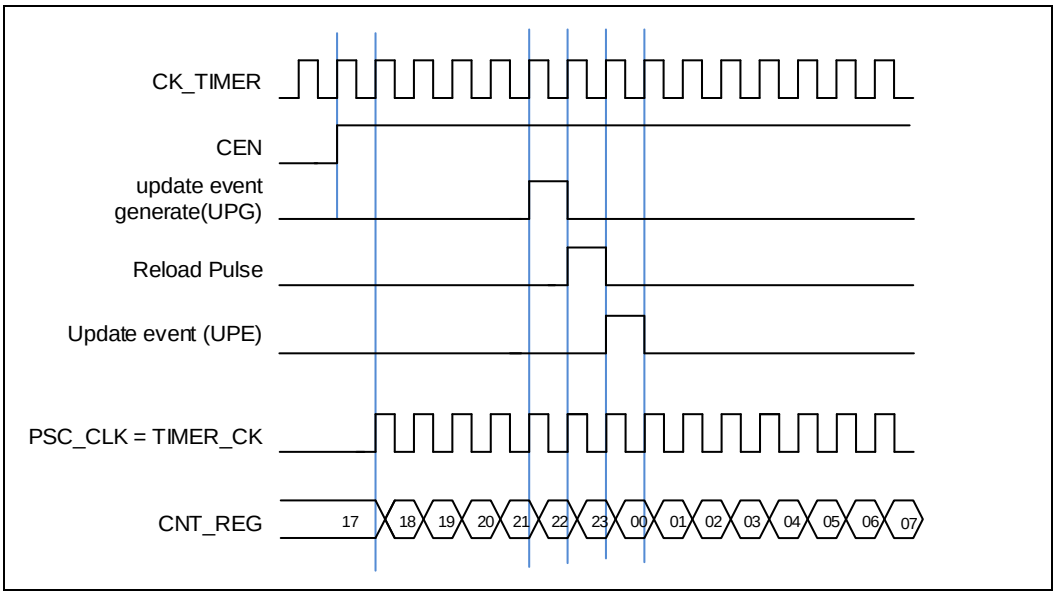
18.5.4. Function overview

Clock source configuration

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.

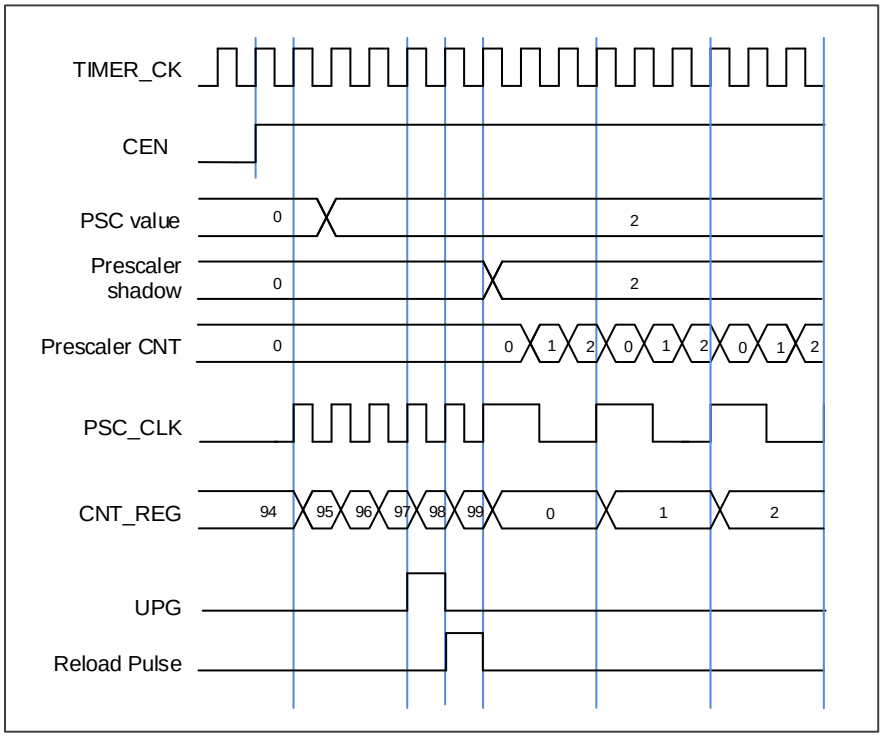
Figure 18-68. Timing chart of internal clock divided by 1



Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 18-69. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again. 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 shadow registers (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=0x99.

Figure 18-70. Timing chart of up counting mode, PSC=0/2

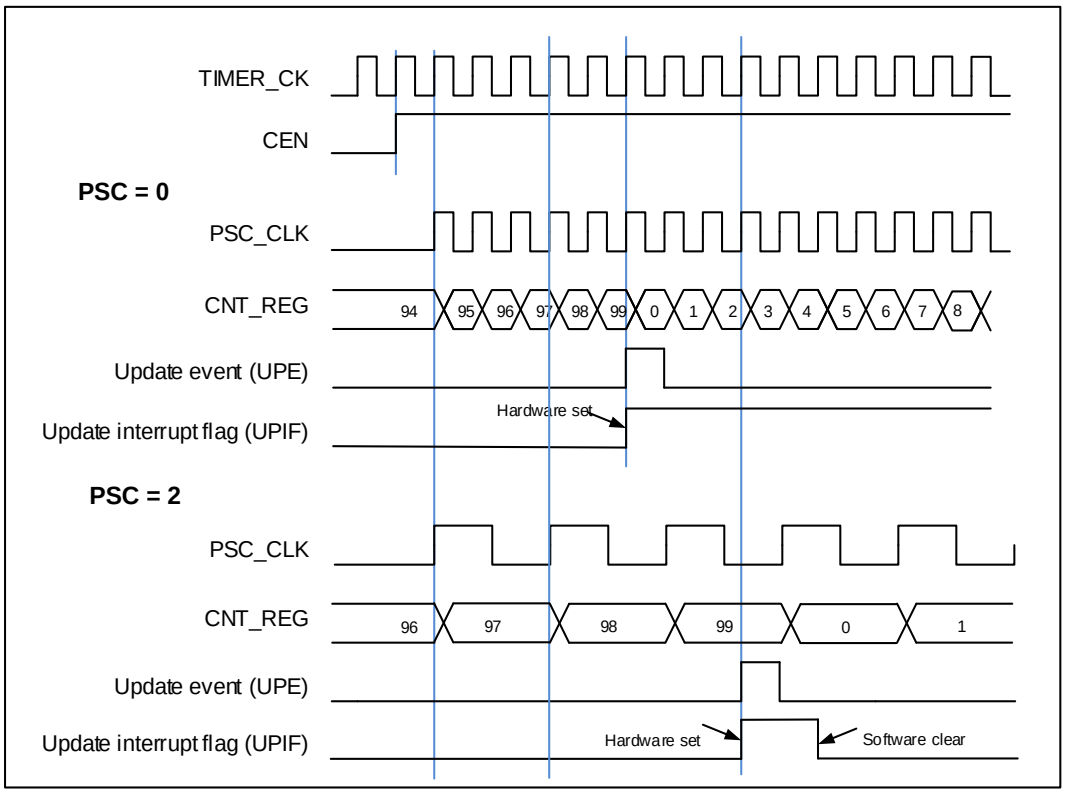
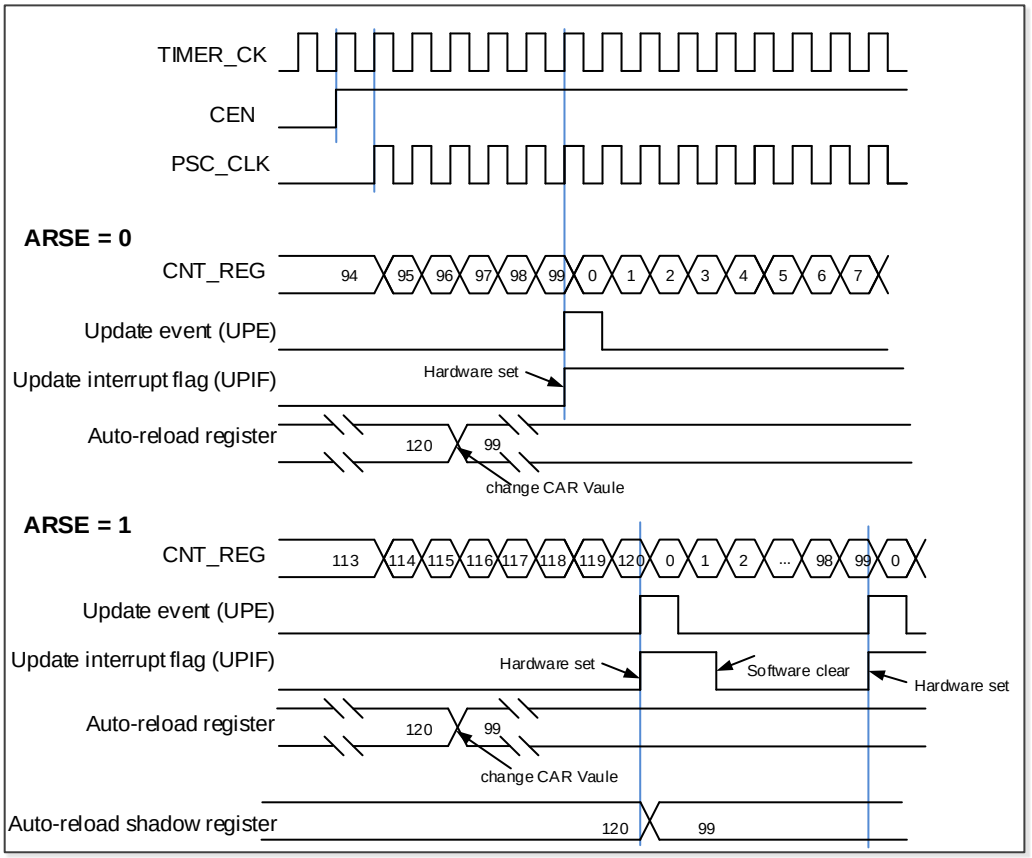


Figure 18-71. Timing chart of up counting mode, change TIMERx_CAR ongoing



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.

Once the timer is set to operate in the single pulse mode, it is necessary to set the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter, then the CEN bit keeps 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.

Timer debug mode

When the Cortex®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL0 register set to 1, the TIMERx counter stops.

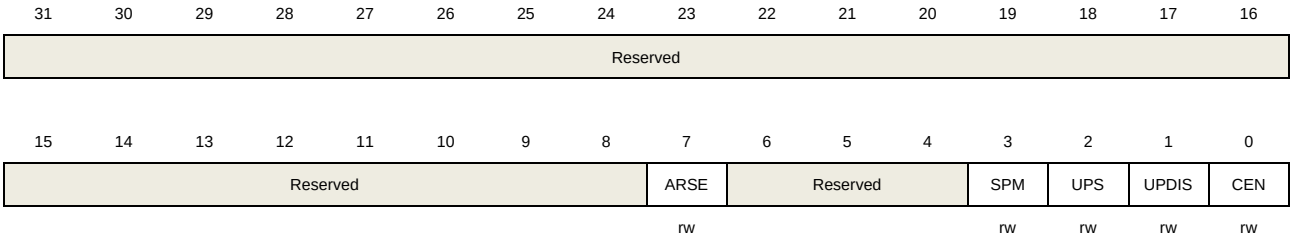
18.5.5. TIMERx registers(x=5, 6)

TIMER5 base address: 0x4000 1000
TIMER6 base address: 0x4000 1400

Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

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



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. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set

The counter generates an overflow or underflow event

The restart mode generates an update event.

1: Update event disable.

Note: When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.

0 CEN Counter enable

0: Counter disable

1: Counter enable

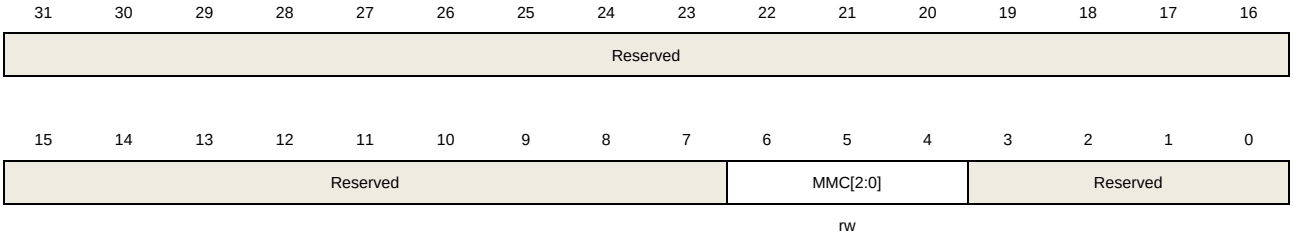
The CEN bit must be set by software when timer works in external clock, pause mode and quadrature decode mode.

Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

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

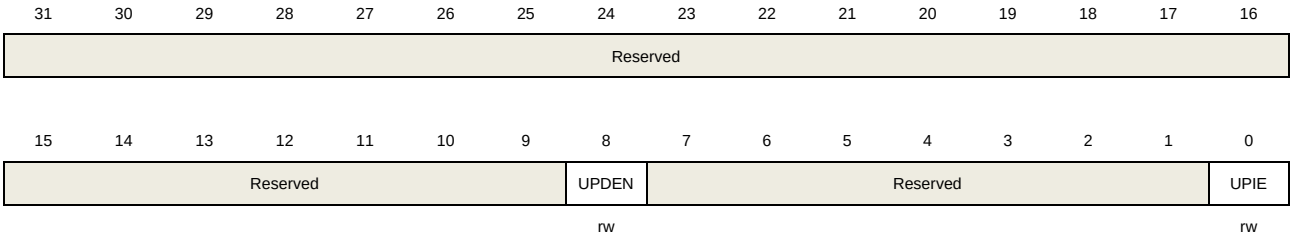


Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6:4	MMC[2:0]	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source: Master timer generate a reset the UPG bit in the TIMERx_SWEVG register is set 001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source : CEN control bit is set The trigger input in pause mode is high 010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.
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).

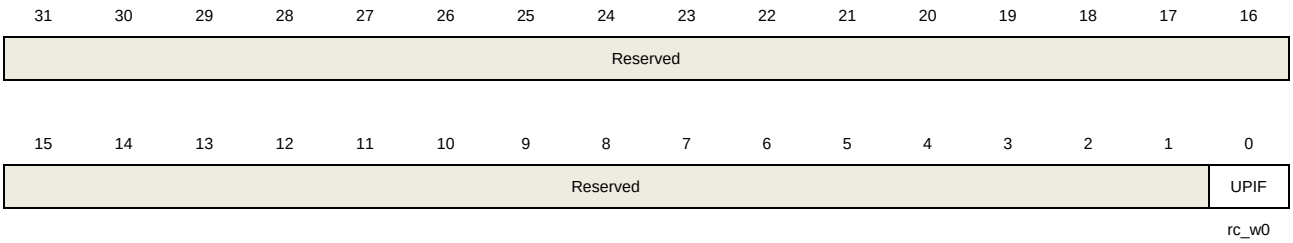


Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7:1	Reserved	Must be kept at reset value.
0	UPIE	Update interrupt enable 0: disabled 1: enabled

Interrupt flag register (TIMERx_INTF)

Address offset: 0x10
Reset value: 0x0000 0000

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



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 TIMER_CK 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.

19. Super High-Resolution Timer (SHRTIMER)

19.1. Overview

SHRTIMER has a super high-resolution counting clock and can be used for high-precision timing. It can generate 10 super high resolution and flexible digital signals to control motor or be used for power management applications. The 10 digital signals can be output independently or coupled into 5 pairs of complementary signals.

It has a flexible capture function and can be used to capture timing the input signal. It has multiple internal signals connected to the ADC and DAC. It can be used for control and monitoring purposes.

It can handle various fault input for safe purposes.

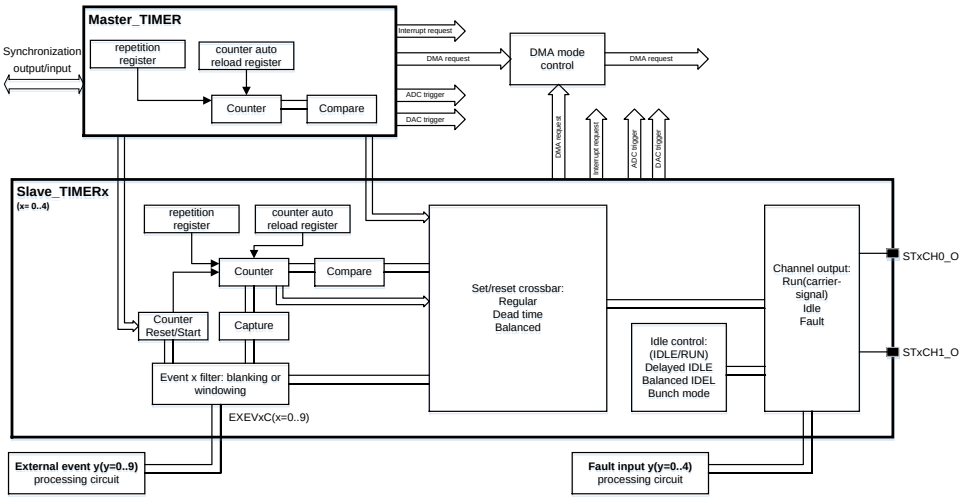
19.2. Characteristics

- Super high- resolution timing units: Master_TIMER, Slave_TIMERx (x=0..4).
- 10 digital signals outputs: they can be controlled by any timing unit and output independently or coupled into 5 pairs.
- Synchronization outputs: synchronize external resources as master.
- Synchronization inputs: be synchronized as a slaver.
- Multiple internal signals connected to the ADC and DAC.
- Various fault input protection scheme: fault input channel and system fault.
- Bunch mode controller to handle light-load operation.
- 7 interrupt vectors: Master_TIMER interrupt, Slave_TIMERx (x=0..4) interrupt, fault interrupts.
- 6 DMA request: Master_TIMER requests, Slave_TIMERx (x=0..4) requests.
- DMA mode for multiple registers update.

19.3. Block diagram

Figure 19-1. SHRTIMER block diagram provides details of the internal configuration of the SHRTIMER timer.

Figure 19-1. SHRTIMER block diagram



19.4. Function overview

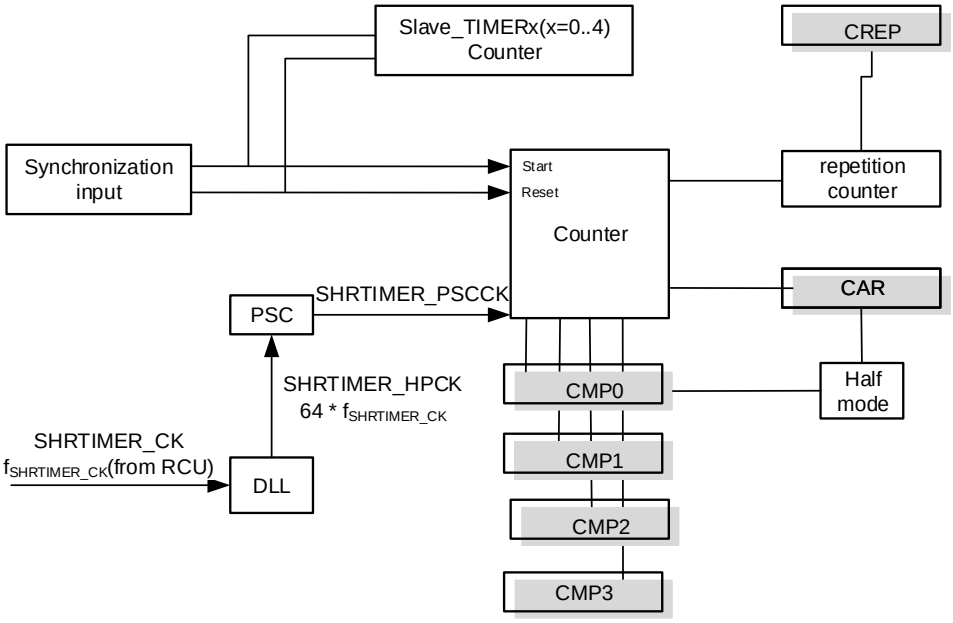
19.4.1. Master_TIMER unit

The Master_TIMER unit is built around the following components:

- 16-bit counter.
- Auto reload register: counting period.
- Repetition counter.
- Compare y (y=0..3) register.

Figure 19-2. Master_TIMER diagram provides details of the internal configuration of the Master_TIMER timer

Figure 19-2. Master_TIMER diagram



The auto-reload register and compare y (y=0..3) register have the following limitations:

- The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$.
- The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$).

Refer to ***Table 19-1. The limitations of auto-reload and compare y (y=0..3) register.***

Table 19-1. The limitations of auto-reload and compare y (y=0..3) register

CNTCKDIV[3:0]	Min value	Max value
4'b1000	0x00C0	0xFFBF
4'b0000	0x0060	0xFFDF
4'b0001	0x0030	0xFFEF
4'b0010	0x0018	0xFFF7
4'b0011	0x000C	0xFFFB
4'b0100	0x0006	0xFFFD
4'b0101	0x0003	0xFFFE
4'b0110	0x0003	0xFFFE
4'b0111	0x0003	0xFFFE

Counter clock

The Master TIMER can have two clock sources, CK_APB2 and CK_SYS, which are provided by the RCU module. If the user selects CK_APB2 to generate SHRTIMER_CK, the CNTCKDIV[2:0] value in the SHRTIMER_MTCTL0 register must be at least 5 (prescaler ratio of 64 or greater);The DLL is used to produce a super high resolution clock SHRTIMER_HPCK ($f_{SHRTIMER\_HPCK}$ = 64 *$f_{SHRTIMER\_CK}$),In this case, SHRTIMER_CK must be generated by CK_SYS.Refer to ***DLL calibrate*** for more information.

When the CNTCKDIV[3] bit in SHRTIMER_MTACTL register is '0', the prescaler (PSC) can

divide the super high resolution clock (SHRTIMER_HPCK) to the counter clock (SHRTIMER_PSCCK) by factor $2^{CNTCKDIV[2:0]+1}$ which is controlled by CNTCKDIV[2:0] bit-field in SHRTIMER_MTCTL0 register. The frequency relationship between them can be expressed below:

$$f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK} / 2^{CNTCKDIV[2:0]+1}$$

When the CNTCKDIV[3] bit in SHRTIMER_MTACTL register is '1', the CNTCKDIV[2:0] bit-filed can only be configured with '3'b000' and the frequency relationship between SHRTIMER_PSCCK and SHRTIMER_HPCK can be expressed below:

$$f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}$$

Note: The clock division CNTCKDIV[3:0] cannot be modified once the Master_TIMER is enabled. CNTCKDIV[3] bit is in the SHRTIMER_MTACTL register and CNTCKDIV[2:0] is in SHRTIMER_MTCTL0 register.

Figure 19-3. Counter clock when divided by 32 shows some behavior of the counter when the register SHRTIMER_MTCAR is set to 0x104 as well as the field CNTCKDIV[3:0] is set to 4'b0100.

Figure 19-3. Counter clock when divided by 32

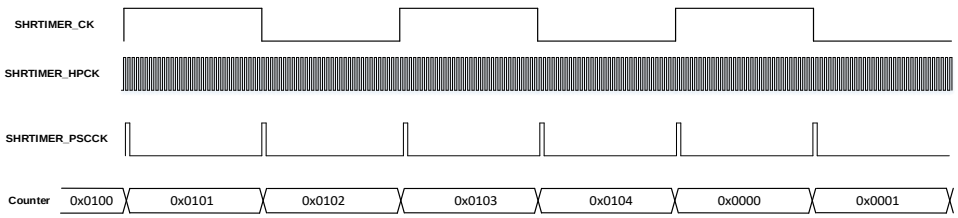


Table 19-2. Resolution with fSHRTIMER_CK = 180MHz shows the various resolutions when the $f_{SHRTIMER\_CK}$ is 180MHz.

Table 19-2. Resolution with $f_{SHRTIMER\_CK}$ = 180MHz

CNTCKDIV[2:0]	$f_{SHRTIMER\_PSCCK}$	Resolution
4'b0000	180*32MHz=5.76GHz	173.6ps
4'b0001	180*16MHz=2.880GHz	347.2ps
4'b0010	180*8MHz=1.440GHz	694.4ps
4'b0011	180*4MHz=720MHz	1.4ns
4'b0100	180*2MHz=360MHz	2.8ns
4'b0101	180*1MHz=180MHz	5.6ns
4'b0110	180/2MHz=90MHz	11.1ns
4'b0111	180/4MHz=45MHz	22.2ns
4'b1000	180*64MHz=11.520G	86.8ps

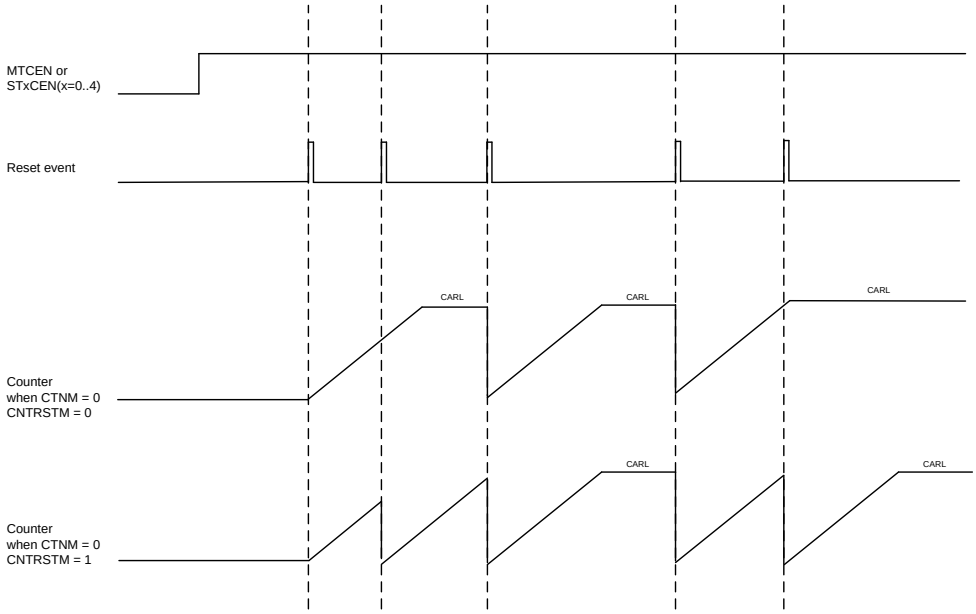
Up counting mode

The counter counts up continuously from 0 to the counter-reload value, which is defined in

the SHRTIMER_MTCAR register. There are two counter operating mode: either single pulse mode (CTNM = 0 in SHRTIMER_MTCTL0 register) or continuous mode (CTNM = 1 in SHRTIMER_MTCTL0 register).

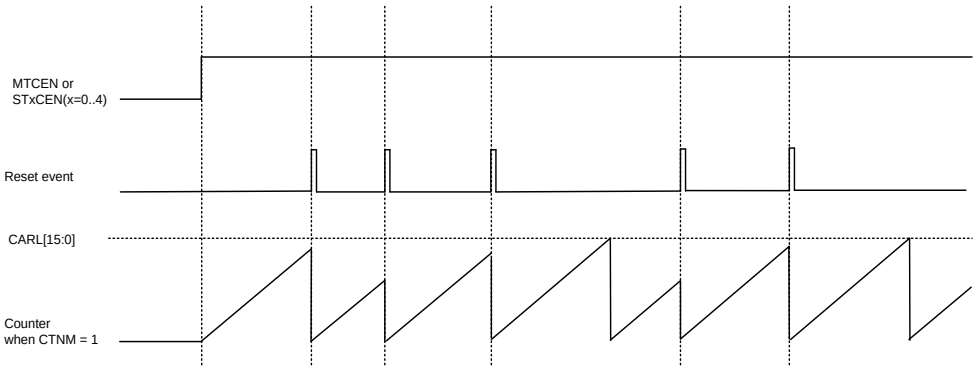
In single pulse mode, after setting the bit MTCEN in SHRTIMER_MTCTL0 register, the first reset event will start the counter. When counting up to the counter-reload value, the counter stops and generates a period event. Then the other reset event will reset and restart the counter. During counting process, the reset event will reset and restart the counter if CNTRSTM = 1 in SHRTIMER_MTCTL0 register, otherwise it will be ignored. ***Figure 19-4. Counter behavior in single pulse mode*** shows the counter operation diagram in single pulse mode.

Figure 19-4. Counter behavior in single pulse mode



In continuous mode, the counter starts immediately as soon as MTCEN bit in SHRTIMER_MTCTL0 register is set to 1. When counting up to the counter-reload value, the counter restarts from 0 and a roll-over event is generated. Different from single pulse mode, the reset events generated at any time will reset and restart the counter. ***Figure 19-5. Counter behavior in continuous mode*** shows counter operation diagram in continuous mode.

Figure 19-5. Counter behavior in continuous mode



Repetition counter

When MTCEN bit in SHRTIMER_MTCTL0 register is set to 1, the repetition counter load the value of SHRTIMER_MTCREP register. The repetition counter is decremented when the counter is cleared due to either a roll-over event in continuous mode or a reset event. When the repetition counter has reached zero, the coming roll-over event in continuous mode or reset event will generate a repetition event and reload the value of SHRTIMER_MTCREP register.

The repetition event will set REPIF bit in SHRTIMER_MTINTF register to 1, and a repetition interrupt or DMA request is issued if enabled (REPIE = 1 or REPDEN = 1 bits in SHRTIMER_MTDMAINTEN register). The repetition interrupt flag can be cleared by writing 1 to REPIFC bit in SHRTIMER_MTINTFC.

Figure 19-6. Repetition counter behavior in continuous mode shows repetition counter operation diagram in continuous mode.

Figure 19-6. Repetition counter behavior in continuous mode

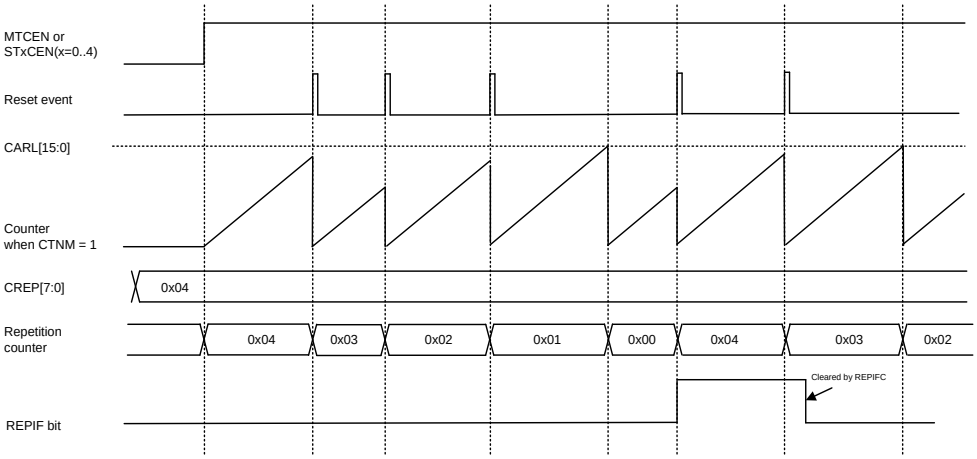


Figure 19-7. Repetition counter behavior in single pulse mode with CNTRSTM = 0 shows repetition counter operation diagram in single pulse mode with CNTRSTM = 0.

Figure 19-7. Repetition counter behavior in single pulse mode with CNTRSTM = 0

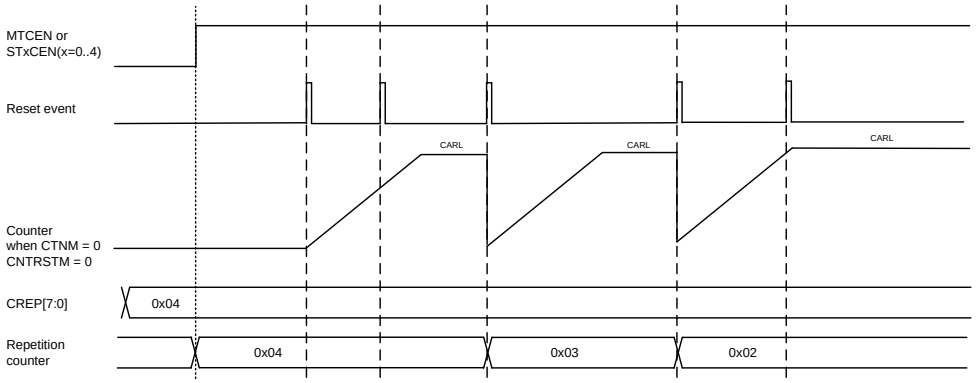
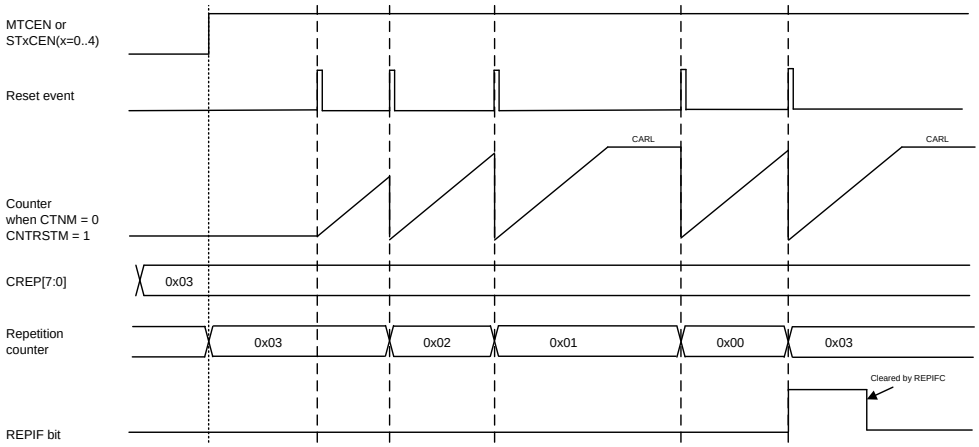


Figure 19-8. Repetition counter behavior in single pulse mode with CNTRSTM = 1 shows repetition counter operation diagram in single pulse mode with CNTRSTM = 1.

Figure 19-8. Repetition counter behavior in single pulse mode with CNTRSTM = 1



Counter reset

The counter can be reset to 0 by software or synchronous input only once the counter is enabled (MTCEN = 1).

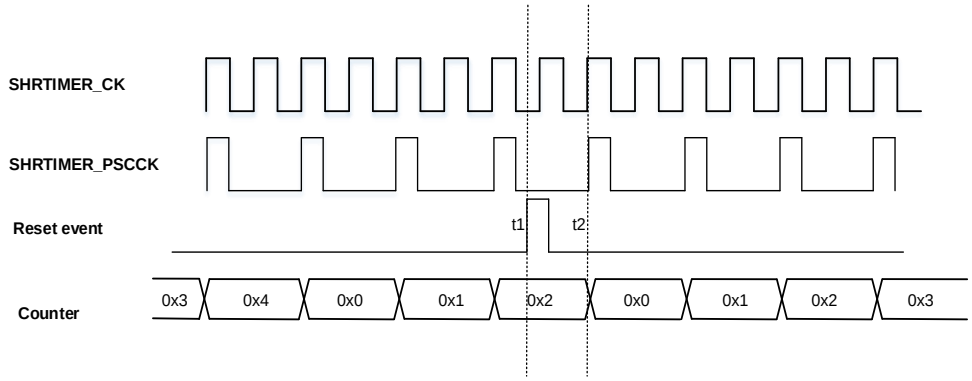
Writing 1 to the MTSRST bit, cleared by hardware automatically, makes the counter reset.

When SYNIRST is set to 1 in SHRTIMER_MTCTL0 register, the synchronous input can reset the counter. The details refer to *Synchronization input*.

When the counter clock SHRTIMER_PSCCK prescaling ratio is above 64 (CNTCKDIV [3]=1'b0 and CNTCKDIV[2:0] > 3'b101), the counter reset event is delayed to the next rising edge of the SHRTIMER_PSCCK.

Figure 19-9 Reset event resynchronization when prescaling ratio is 128 show CNTCKDIV[3:0] = 4'b0110 in continuous mode with SHRTIMER_MTCAR = 0x4.

Figure 19-9 Reset event resynchronization when prescaling ratio is 128



Compare

The Master_TIMER unit has four compare registers: SHRTIMER_MTCMPxV(x=0..3). When the counter value matches the compare registers value, a coincident compare event is generated.

The compare event will set the corresponding compare interrupt flag to 1 (CMPxIF bit in SHRTIMER_MTINTF where x=0..3), and a compare interrupt or DMA request is issued if enabled (CMPxIE = 1 or CMPxDEN = 1 bits in SHRTIMER_MTDMAINTEN register where x=0..3). The compare interrupt flag can be cleared by writing 1 to CMPxIFC bit in SHRTIMER_MTINTFC where x=0..3.

Half mode

When HALFM bit in SHRTIMER_MTCTL0 is set 1, the half mode is enabled. This mode forces the value of compare 0 active register to be half of the counter-reload value, but the value of SHRTIMER_MTCMP0V register is not updated with the SHRTIMER_MTCAR / 2 value. It is mainly used to generate a square wave with a fixed duty cycle of 50%.

When SHWEN bit in SHRTIMER_MTCTL0 is set to 1, the shadow registers are enabled and the compare 0 active register is refreshed on the update event. Otherwise, the compare 0 active register is refreshed as soon as the new value is written into SHRTIMER_MTCAR.

Synchronization input start/reset counter

Synchronous input can generate a counter reset event when SYNIRST is set to 1 in SHRTIMER_MTCTL0 register and start counter when SYNISTRT is set to 1 in SHRTIMER_MTCTL0 register. Refer to ***Synchronization input*** for more information.

A synchronization input request will set the SYNIIF bit in SHRTIMER_MTINTF register to 1, and a interrupt or a DMA request is issued if enabled(SYNIIE = 1 or SYNIDEN = 1 bits in SHRTIMER_MTDMAINTEN register).The synchronization input interrupt flag can be cleared by writing 1 to SYNIIFC bit in SHRTIMER_MTINTFC.

Update event and shadow registers

Some registers in Master_TIMER contain shadow registers. The shadow registers are disabled after MCU reset.

If the SHWEN bit in SHRTIMER_MTCTL0 register is cleared to 0, shadow registers are disabled. The values written into these registers are transferred into active register and take effect immediately.

If the SHWEN bit in SHRTIMER_MTCTL0 register is set to 1, shadow registers are enabled and these registers are preloaded. The values written into these registers are transferred into the shadow register and do not take effect immediately. Their content of the shadow is transferred into active register and take effect immediately on update event.

Note: Update event occurs only when SHWEN=1.

Table 19-3. Master_TIMER shadow registers and update event lists the registers containing shadow registers and the relevant event.

Table 19-3. Master_TIMER shadow registers and update event

Registers that contain shadow registers	Shadow registers enable bit	Update event.
SHRTIMER_MTDMAINTEN	SHWEN bit in SHRTIMER_MTCTL0 register	Software(MTSUP bit)
SHRTIMER_MTCAR		Repetition event(UPREP = 1)
SHRTIMER_MTCREP		DMA mode end event(UPSEL[1:0] = 2'b01)
SHRTIMER_MTCMP0V		Roll-over event following a DMA mode end event (UPSEL[1:0] = 2'b10)
SHRTIMER_MTCMP1V		
SHRTIMER_MTCMP2V		
SHRTIMER_MTCMP3V		

The Master_TIMER has 4 update options:

1. Software generates an update event. Writing 1 to MTSUP bit in SHRTIMER_CTL1 register can generate an update event and cancel all the pending hardware update events, regardless of how UPSEL[1:0] in SHRTIMER_MTCTL0 register is configured

2. A repetition event generates an update event. If UPREP bit is set to 1 in SHRTIMER_MTCTL0 register, the repetition event due to either roll-over or reset events can generate an update event. If UPSEL[1:0]=2'b10 in SHRTIMER_MTCTL0 register, the repetition event can't generate update event.

3. An update event generated when the DMA transfer is completed in DMA mode. If UPSEL[1:0]=2'b01 in SHRTIMER_MTCTL0 register, An update event is automatically generated by the hardware when the DMA transfer is completed in DMA mode. It is also possible to generate update event by software or repetition event.

4. Update event generated on counter roll-over following a DMA transfer completion in DMA mode. If UPSEL[1:0]=2'b10 in SHRTIMER_MTCTL0 register, An update event is

automatically generated by the hardware when a counter roll-over event following a DMA transfer completion is generated in DMA mode. It is also possible to generate update event by software.

A update event will set UPIF bit in SHRTIMER_MTINTF register to 1, and an interrupt or a DMA request is issued if enabled(UPIE = 1 or UPDEN = 1 bits in SHRTIMER_MTDMAINTEN register).The update event interrupt flag can be cleared by writing 1 to UPIFC bit in SHRTIMER_MTINTFC.

DAC Trigger

When the Master_TIMER update event occurs, a DAC trigger request can be generated on SHRTIMER_DACTRIGOy(y=0..2) if DACTRGS[1:0] != 2'b00 in SHRTIMER_MTCTL0 register. If DACTRGS[1:0] = 2'b00 in SHRTIMER_MTCTL0 register, it won't generate DAC trigger request. SHRTIMER_DACTRIGOy(y=0..2) is the internal signal connected from Master_TIMER to the DAC module. Refer to ***Trigger to DAC*** for more information.

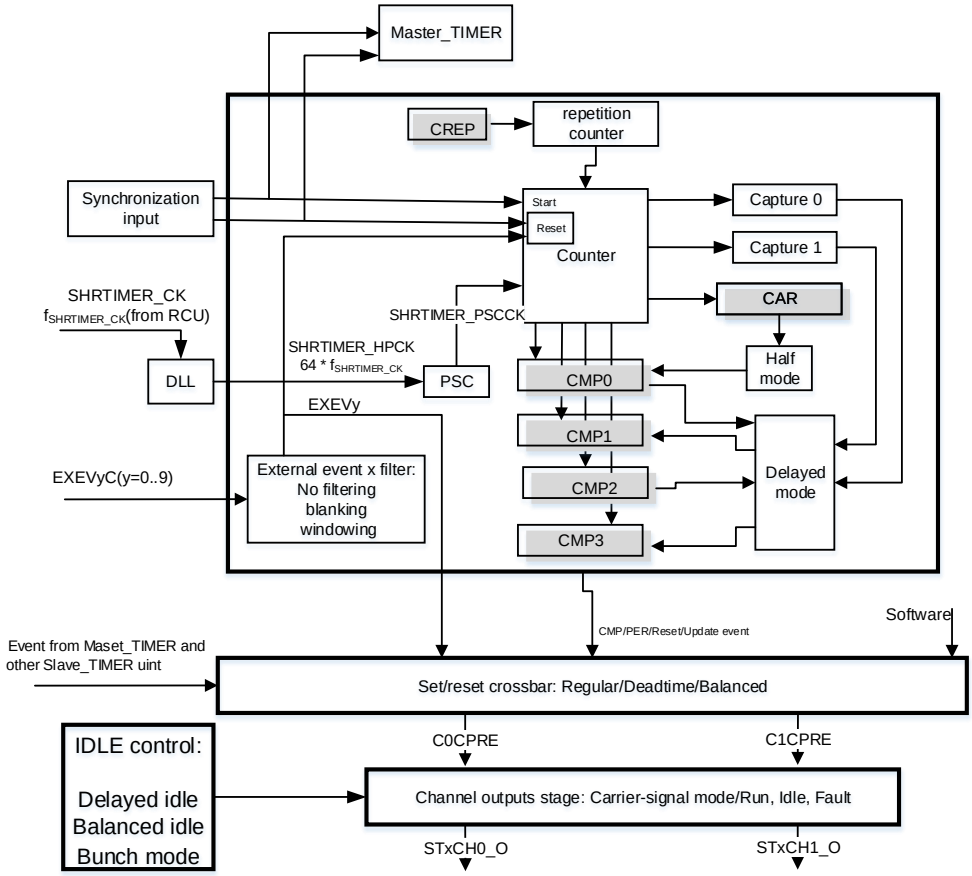
19.4.2. Slave_TIMERx(x=0..4) unit

The SHRTIMER has 5 slave timers with similar structure: Slave_TIMERx(x=0..4). Each unit is built around the following components:

- 16-bit counter.
- Auto reload register: counting period.
- Repetition counter.
- Compare y (y=0..3) register.
- Capture y(y=0,1) register.
- Set/reset crossbar.
- IDEL control stage
- Channel output stage.

Figure 19-10. Slave_TIMERx diagram shows the structure of Slave_TIMERx.

Figure 19-10. Slave_TIMERx diagram



The auto-reload register and compare y (y=0..3) registers have the following limitations:

- The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$.
- The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$).

Refer to ***Table 19-1. The limitations of auto-reload and compare y (y=0..3) register.***

The counter and capture y(y=0,1) value registers also have the following limitations: for counter clock division below 64 (CNTCKDIV[3:0] < 4'b0101 or CNTCKDIV[3:0] =4'b1000), the least significant bits are not significant. They cannot be written and read 0. Refer to ***Table 19-4. The limitations of counter and capture y(y=0,1) value registers.***

Table 19-4. The limitations of counter and capture y(y=0,1) value registers

CNTCKDIV[3:0]	Bits which are not significant
4'b1000	Bit5 to bit0
4'b0000	Bit4 to bit0
4'b0001	Bit3 to bit0
4'b0010	Bit2 to bit0
4'b0011	Bit1 to bit0
4'b0100	Bit0
4'b0101	x
4'b0110	x

4'b0111	x

Note: "x" means that all bits are significant.

Counter clock

The Slave TIMER can have two clock sources, CK_APB2 and CK_SYS, which are provided by the RCU module. If the user selects CK_APB2 to generate SHRTIMER_CK, the CNTCKDIV[2:0] value in the SHRTIMER_STxCTL0 register must be at least 5 (prescaler ratio of 64 or greater)；The DLL is used to produce a super high resolution clock SHRTIMER_HPCK ($f_{SHRTIMER\_HPCK} = 64 * f_{SHRTIMER\_CK}$), In this case, SHRTIMER CK must be generated by CK_SYS. Refer to ***DLL calibrate*** for more information.

When the CNTCKDIV[3] bit in SHRTIMER_STxACTL register is '0', the prescaler (PSC) can divide the super high resolution clock (SHRTIMER_HPCK) to the counter clock (SHRTIMER_PSCCK) by factor $2^{CNTCKDIV[2:0]+1}$ which is controlled by CNTCKDIV[2:0] bit-field in SHRTIMER_MTCTL0 register. The frequency relationship between them can be expressed below:

$$f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK} / 2^{CNTCKDIV[2:0]+1}$$

When the CNTCKDIV[3] bit in SHRTIMER_MTACTL register is '1' ,the CNTCKDIV[2:0] bit-filed can only be configured with '3'b000' and the frequency relationship between SHRTIMER_PSCCK and SHRTIMER_HPCK can be expressed below:

$$f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}$$

Note: The clock division CNTCKDIV[3:0] cannot be modified once the Slave_TIMERx is enabled. CNTCKDIV[3] bit is in the SHRTIMER_STxACTL register and CNTCKDIV[2:0] is in SHRTIMER_STxCTL0 register.

Refer to ***Figure 19-3. Counter clock when divided by 32*** and ***Table 19-2. Resolution with fSHRTIMER_CK = 180MHz*** for more information.

Up counting mode

The counter counts up continuously from 0 to the counter-reload value, which is defined in the SHRTIMER_STxCAR register. There are two counter operating modes: single pulse mode (CTNM = 0 in SHRTIMER_STxCTL0 register) and continuous mode (CTNM = 1 in SHRTIMER_STxCTL0 register).

In single pulse mode, after setting the bit STxCEN in SHRTIMER_MTCTL0 register, the first reset event will start the counter. When counting up to the counter-reload value, the counter stops and generates a period event. Then the other reset event will reset and restart the counter. During counting process, the reset event will reset and restart the counter if CNTRSTM = 1 in SHRTIMER_STxCTL0 register, otherwise it will be ignored. Refer to ***Figure 19-4. Counter behavior in single pulse mode***.

In continuous mode, the counter starts immediately as soon as STxCEN bit in

SHRTIMER_MTCTL0 register is set to 1. When counting up to the counter-reload value, the counter restarts from 0 and a roll-over event is generated. Different from single pulse mode, the reset events generated at any time will reset and restart the counter. Refer to ***Figure 19-5. Counter behavior in continuous mode***.

Repetition counter

When STxCEN bit in SHRTIMER_MTCTL0 register is set to 1, the repetition counter load the value of SHRTIMER_STxCREP register. The repetition counter is decremented when the counter is cleared due to either a roll-over event in continuous mode or a reset event. When the repetition counter has reached zero, the coming roll-over event in continuous mode or reset event will generate a repetition event and reload the value of SHRTIMER_STxCREP register.

The repetition event will set REPIF bit in SHRTIMER_STxINTF register to 1, and a repetition interrupt or DMA request is issued if enabled (REPIE = 1 or REPDEN = 1 bits in SHRTIMER_STxDMAINTEN register).The repetition interrupt flag can be cleared by writing 1 to REPIFC bit in SHRTIMER_STxINTFC.

The repetition counter behavior in continuous mode refer to ***Figure 19-6. Repetition counter behavior in continuous mode***.

The repetition counter behavior in single pulse mode with CNTRSTM = 0 refer to ***Figure 19-7. Repetition counter behavior in single pulse mode with CNTRSTM = 0***.

The repetition counter behavior in single pulse mode with CNTRSTM = 1 refer to ***Figure 19-8. Repetition counter behavior in single pulse mode with CNTRSTM = 1.***

Counter reset

The counter can be reset by three types of signal sources:

1. Software. Writing 1 to the STxSRST bit in SHRTIMER_CTL1 register.
2. Synchronization input start/reset counter.
3. Events configured in SHRTIMER_STxCNTRST register.

All these sources are logical ORed, they can be valid simultaneously. If multiple reset events occur in the same $t_{SHRTIMER\_CK}$ cycle, only the last one is valid. The counter reset requests are taken into account only once the related Slave_TIMERx are enabled

Note: If the external events is configured with level sensitivity, only one external events can be enabled in the SHRTIMER_STxCNTRST register.

Writing 1 to the STxSRST bit, cleared by hardware automatically, makes the counter reset. These control bits of Master_TIMER and Slave_TIMERx(x=0..4) are grouped into SHRTIMER_CTL1 register to allow the simultaneous reset of several counters.

When SYNIRST is set to 1 in SHRTIMER_STxCTL0 register, the synchronous input can reset

the counter. Refer to ***Synchronization input***.

There are 30 events that can be selected simultaneously to reset counter in the SHRTIMER_STxCNTRST register and these can be classified into four categories:

- Slave_TIMERx itself: update event, compare 1 event and compare 3 event.
- Other Slave_TIMERy (for instance x=1, then y=0, 2..4) : compare 0 event, compare 1 event and compare 3 event.
- Master_TIMER: compare 0 event, compare 1 event, compare 2 event, compare 3 event and reset event.
- External event y(y=0..9): EXEVy conditioned by external event filter in Slave_TIMERx

When the counter clock SHRTIMER_PSCCK prescaling ratio is above 64 (CNTCKDIV [3]=1'b0 and CNTCKDIV[2:0] > 3'b101), the counter reset event is delayed to the next rising edge of the SHRTIMER_PSCCK. The details refer to ***Figure 19-9 Reset event resynchronization when prescaling ratio is 128.***

The counter reset event will set RSTIF bit in SHRTIMER_STxINTF register to 1, and a counter reset interrupt or DMA request is issued if enabled(RSTIE = 1 or RSTDEN = 1 bits in SHRTIMER_STxDMAINTEN register).The counter reset interrupt flag can be cleared by writing 1 to RSTIFC bit in SHRTIMER_STxINTFC.

Capure

Capture feature not only allows the Slave_TIMERx to perform measurements such as pulse interval, frequency, period, duty cycle and so on, but also to update compare 1 and compare 3 values in delayed mode (refer to ***Delayed mode***).

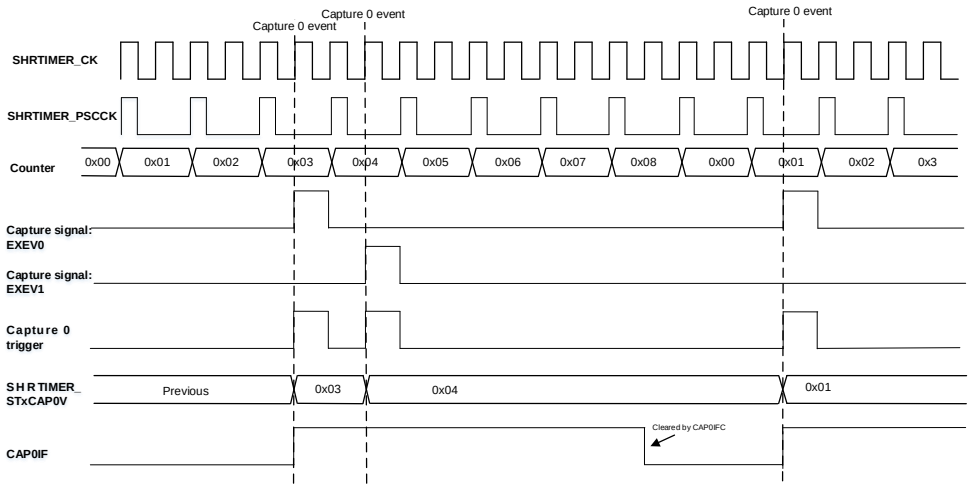
When a selected trigger signal occurs, the current value of the counter is captured into the SHRTIMER_STxCAPyV(y=0,1) register. At the same time the CAPyIF(y=0,1) bit in SHRTIMER_STxINTF register is set and the capture interrupt and DMA request is generated if enabled by CAPyIE(y=0,1) = 1 and CAPyDEN(y=0,1) = 1 in SHRTIMER_STxDMAINTEN register. The capture interrupt flag CAPyIF can be cleared by writing 1 to CAPyIFC bit in SHRTIMER_STxINTFC.

The capture 0 trigger events are defined in SHRTIMER_STxCAP0TRG register and the capture 1 trigger events are defined in SHRTIMER_STxCAP1TRG register. All the trigger events are logical ORed and they are all valid when multiple trigger events are selected.

Note: If the external events is configured with level sensitivity, only one external events can be enabled in the SHRTIMER_STxCAPyTRG(y=0,1) register.

The over-capture is not prevented. Even if the previously captured value is not read or the capture flag is not cleared, the new capture will still be triggered and the new captured value will override the previous value. Refer to ***Figure 19-11. Capture 0 triggered by EXEV0 and EXEV1***.

Figure 19-11. Capture 0 triggered by EXEV0 and EXEV1

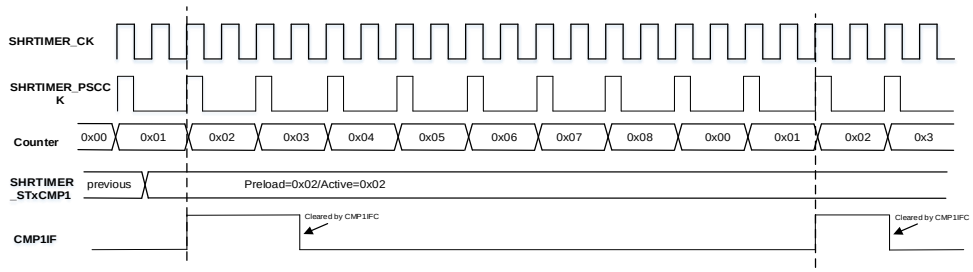


Compare

The Slave_TIMERx unit has four compare registers: SHRTIMER_STxCMPyV(y=0..3). When the counter value matches the compare registers value, a coincident compare event is generated. Refer to ***Figure 19-12 . Compare 1 behavior with STxCAR=0x8, STxCMP1V=0x02***.

The compare event will set the corresponding compare interrupt flag to 1 (CMPyIF bit in SHRTIMER_STxINTF where y=0..3), and a compare interrupt or DMA request is issued if enabled (CMPyIE = 1 or CMPyDEN = 1 bits in SHRTIMER_STxDMAINTEN register where y=0..3). The compare interrupt flag can be cleared by writing 1 to CMPyIFC bit in SHRTIMER_STxINTFC.

Figure 19-12. Compare 1 behavior with STxCAR=0x8, STxCMP1V=0x02



Half mode

When HALFM bit in SHRTIMER_STxCTL0 is set to 1, the half mode is enabled. This mode forces the value of compare 0 active register to be half of the counter-reload value, but the value of SHRTIMER_STxCMP0V register is not updated with the SHRTIMER_MTCAR/2 value. It is mainly used to generate a square wave with a fixed duty cycle of 50%.

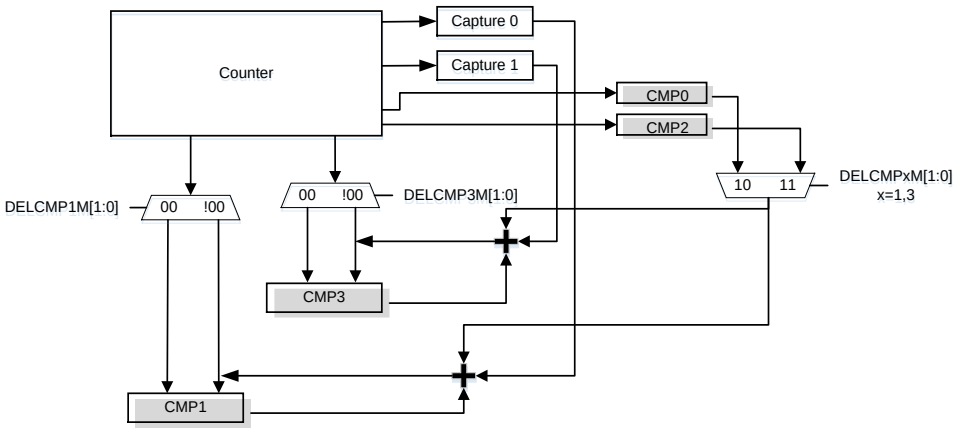
When SHWEN bit in SHRTIMER_STxCTL0 is set to 1, the shadow registers are enabled and the compare 0 active register is refreshed on the update event. Otherwise, the compare 0

active register is refreshed as soon as the new value is written into SHRTIMER_STxCAR.

Delayed mode

This mode is available only for compare y(y=1,3) and controlled by DELCMPyM[1:0] bit-field in SHRTIMER_STxCTL0 register. The actual value compared to the counter is the recalculated value which is recomputed following a capture0/1 trigger or compare0/2 event, as shown in ***Figure 19-13. Compare delayed mode chart***. This mode allows the generation of waveforms to synchronize with the capture trigger by hardware.

Figure 19-13. Compare delayed mode chart



In delay mode, the compare y(y=1, 3) event is valid from the relative capture/compare event up to the period event. When the counter has reached the period value, the compare y(y=1, 3) event is disabled until a new capture/compare comes.

When no relative capture0/1 trigger or compare0/2 event occurs, no compare y event is generated. Once the relative capture is triggered, the value in compare y active register is summed with the relative SHRTIMER_STxCAP0V/ SHRTIMER_STxCAP1V, and it is compared to the counter. Compare 1 is associated with capture 0 and compare 0/2, while compare 3 is associated with capture 1 and compare 0/2.

Note: The recalculated value is transferred to an internal register which cannot read.

The DELCMP3M[1:0] (compare 3) and DELCMP1M[1:0] (compare 1) in SHRTIMER_STxCTL0 register can be used to configure the delay mode. Take DELCMP1M[1:0] for example：

- 2’b00, compare 1 delayed mode disable.

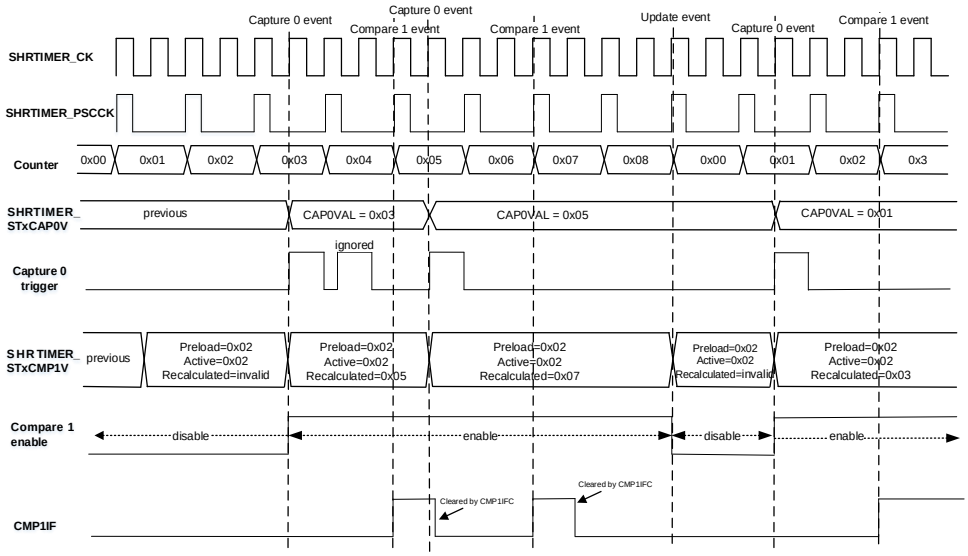
Compare 1 delayed mode disable. Compare match occurs as soon as the counter value is equal to the value of compare 1 active register. Refer to ***Figure 19-12. Compare 1 behavior with STxCAR=0x8, STxCMP1V=0x02***.

- 2’b01, compare 1 delayed mode 0.

After a capture 0 event, the value of compare 1 register is recalculated (compare 1 active

register value + capture 0 value). Compare 1 event occurs as soon as the counter value is equal to the recomputed compare 1 value. Refer to ***Figure 19-14. Compare 1 delayed mode 0.***

Figure 19-14. Compare 1 delayed mode 0



- 2'b10, compare 1 delayed mode 1.

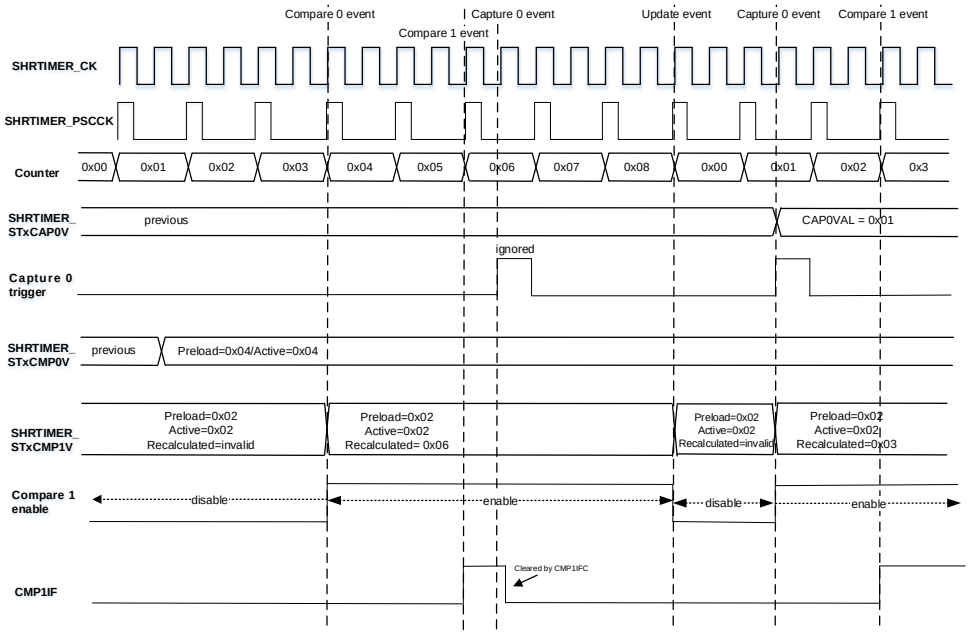
After a capture 0 event or compare 0 event, the value of compare 1 register is recalculated.

For capture 0 trigger, the recomputed value of compare 1 register = compare 1 active register value + capture 0 value for capture 0 event

For compare 0 event, the recomputed value of compare 1 register = compare 1 active register value previous + compare 0 active register value.

Compare 1 event occurs as soon as the counter value is equal to the recomputed compare 1 value. If capture 0 trigger occurs first, the later compare 0 event is ignored. If compare 0 event occurs first, the capture 0 trigger after the compare 1 event is ignored. Refer to ***Figure 19-15. Compare 1 delayed mode 1.***

Figure 19-15. Compare 1 delayed mode 1



- 2'b11, compare 1 delayed mode 2.

This mode is similar to compare 1 delayed mode 1.

After a capture 0 event or compare 2 event, the value of compare 1 register is recalculated.

For capture 0 event, the recalculated value of compare 1 register = compare 1 active register value + capture 0 value for capture 0 event

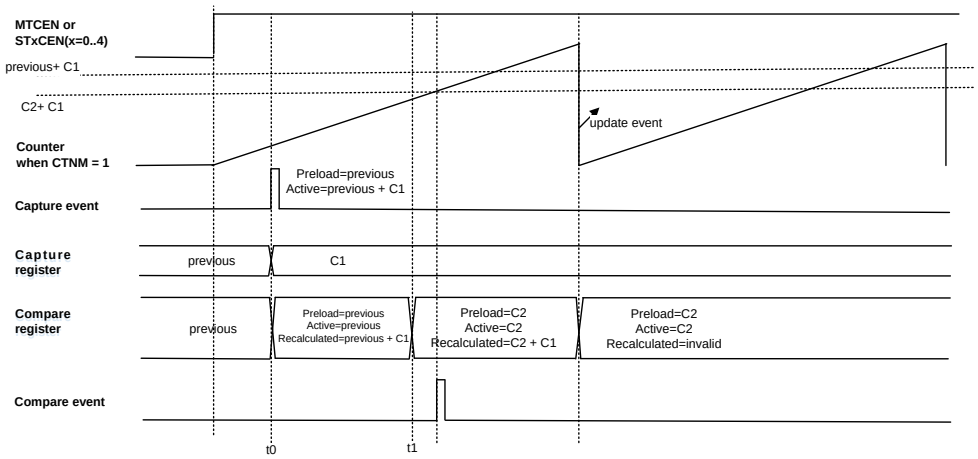
For compare 2 event, the recalculated value of compare 1 register = compare 1 active register value + compare 2 active register value.

Compare match occurs as soon as counter equal the recalculated value of compare 1 register. The captures are not taken into account if they are triggered after (compare 1 active register value + compare 2 active register value). Refer to ***Figure 19-15. Compare 1 delayed mode 1***

When the shadow registers are disabled (SHWEN = 0), the new compare value is taken into the active registers immediately, even if the SHRTIMER_STxCMP0V or SHRTIMER_STxCMP2V are modified after the capture event has occurred. ***Figure 19-16. Compare delayed mode with SHWEN = 0*** shows an example:

At t0, a capture event occurs and C1 is captured into the register. The recalculated value = the value of compare active register + C1. At t1, new compare value (C2) is written into compare register and the recalculated value = C2 + C1.

Figure 19-16. Compare delayed mode with SHWEN = 0



When the delayed mode is enabled (DELCMPyM[1:0]=01/10/11,y=1,3), the over-capture is prevented. In the same counting cycle (determined by SHRTIMER_STxCAR), only the first capture event is taken into account. New capture event are effective in three cases as follows:

- When the recalculated value of compare register matches the counter value.
- When the period event occurs.
- When the counter is reset.

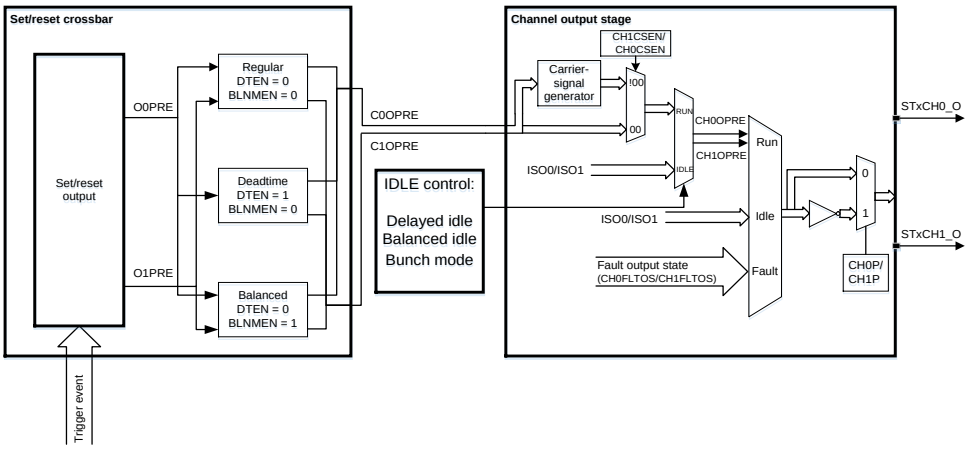
Set/reset crossbar

The channel output waveform can be divided into three stages:

- Set/reset crossbar.
- IDLE control.
- Channel output stage.

Figure 19-17. Channel output diagram shows the structure of the three stages.

Figure 19-17. Channel output diagram



The crossbar has three output modes: regular mode, dead-time mode and balanced mode. Only one of them can be chosen.

Output prepare signal

Slave_TIMERx has a set/reset output module. The module can generate two output prepare signals: O0PRE and O1PRE. O0PRE is controlled by SHRTIMER_STxCH0SET and SHRTIMER_STxCH0RST registers. O1PRE is controlled by SHRTIMER_STxCH1SET and SHRTIMER_STxCH1RST registers. The high level of OyPRE(y=0,1) is active level, while the low level is inactive level.

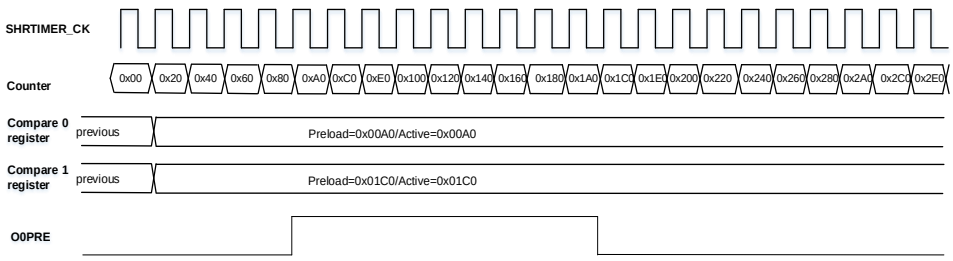
When the event configured in SHRTIMER_STxCHySET register occurs, this module produces a "set request" and make the OyPRE high. When the event configured in SHRTIMER_STxCHyRST occurs, this module produces a "reset request" and make the OyPRE low. If the same event is configured in SHRTIMER_STxCHySET and SHRTIMER_STxCHyRST registers, this module produces a "toggle request" and make the OyPRE toggle the output when the configured event occurs.

Note: If the CNTCKDIV[3:0] field in the SHRTIMER_STxCTL0 and SHRTIMER_STxACTL registers equals to '4'b0110' or '4'b0111', the same event in SHRTIMER_STxCH0SET and SHRTIMER_STxCH0RST registers must not be set simulaneously.

Figure 19-18. O0PRE wave: set on CMP0, reset on CMP1 shows O0PRE wave with the following configuration:

- CNTCKDIV[3:0] = 4'b0000 in SHRTIMER_STxCTL0 and SHRTIMER_STxACTL registers.
- SHRTIMER_STxCH0SET = 0x0000 0008: compare 0 event produces "set request" and O0PRE will output high level.
- SHRTIMER_STxCH0RST = 0x0000 0010: compare 1 event produces "reset request" and O0PRE will output low level.
- SHRTIMER_STxCMP0V = 0x00A0
- SHRTIMER_STxCMP1V = 0x01C0

Figure 19-18. O0PRE wave: set on CMP0, reset on CMP1



There are up to 32 events that can be selected for OyPRE(y=0,1):

- Slave_TIMERx itself: update event, reset event, period event and compare y(y=0..3) event.
- Master_TIMER: period event and compare y(y=0..3) event.

- Slave_TIMERx interconnection event: there are 9 interconnect events from other Slave_TIMERy (for instance x=1, then y=0, 2..4). Refer to ***Table 19-5. Slave_TIMER interconnection event***
- External event y(y=0..9): EXEVy conditioned by external event filter in Slave_TIMERx
- Software event.

Software event is always valid regardless of whether the STxCEN bit is 1. But the other events are only considered when the STxCEN is 1.

Table 19-5. Slave_TIMER interconnection event

Inter-connection	From ST0				From ST1				From ST2				From ST3				From ST4			
	CMP0	CMP1	CMP2	CMP3	CMP0	CMP1	CMP2	CMP3	CMP0	CMP1	CMP2	CMP3	CMP0	CMP1	CMP2	CMP3	CMP0	CMP1	CMP2	CMP3
To ST0	×	×	×	×	0	1	×	2	×	3	×	×	5	6	×	×	×	×	7	8
To ST1	0	×	×	×	×	×	×	×	×	×	3	4	×	×	5	6	7	×	×	×
To ST2	×	×	1	×	×	2	3	×	×	×	×	×	×	4	×	5	×	6	7	8
To ST3	0	×	×	1	×	2	×	3	4	×	5	6	×	×	×	×	7	×	×	8
To ST4	×	×	0	1	×	×	2	3	4	5	×	×	6	7	×	8	×	×	×	×

Note: (1) The numbers in the table represent the Slave_TIMERx interconnection event.
(2) "×" means not available.

Multiple event sources can be selected simultaneously (logic ORed) and when they occur in the same t_{HPTMER_CK} period, an arbitration is performed.

Arbitration mechanism

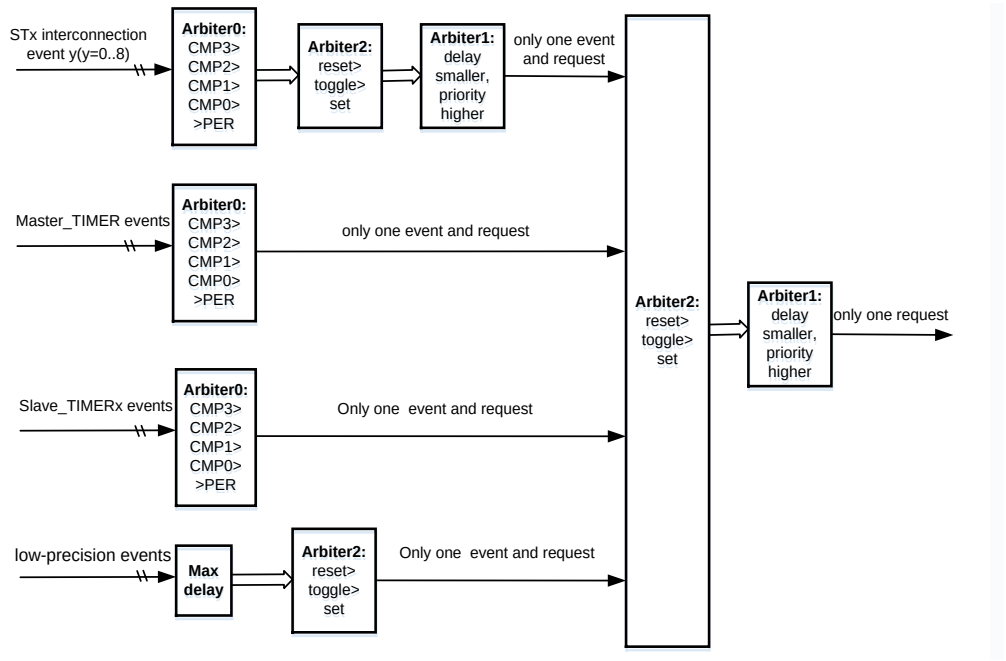
When multiple event configured in SHRTIMER_STxCH1SET and SHRTIMER_STxCH1RST registers occur during a t_{HPTMER_CK} period, an arbitration is performed and only one will be valid to change the OyPRE(y=0,1).

These 32 events can be divided into four types:

- From Slave_TIMERx itself: period event and compare y(y=0..3) event.
- From Master_TIMER: compare y(y=0..3) event, period event.
- From Slave_TIMERx interconnection event: interconnect event y(y=0..8)
- Low-precision events: Slave_TIMERx update event and reset event, external event y(y=0..9), software event.

The arbitration process for each type of event is shown in ***Figure 19-19. Arbitration mechanism during each tSHRTMER_CK period***

Figure 19-19. Arbitration mechanism during each $t_{SHRTMER\_CK}$ period



The functions of the three arbiters are as follows:

- The priority order of arbiter 0(from the highest to the lowest priority):

  Compare 3 event > compare 2 event > compare 1 event > compare 0 event > period event.

- Arbitrator 1 arbitrates priority according to the delay of the event during t_{HPTMER_CK}:
  The smaller the delay, the higher the priority.

- Arbitrator 2 arbitrates priority according to the effect of the event on the OyPRE(y=0,1):

  Reset request > toggle request > set request.

Take O0PRE in Slave_TIMER0 for example and the configuration is:

- SHRTIMER_STxCH0SET = 0x0060 5898, and selected events producing “set request” are:

  From Master_TIMER: compare 2 event, period event.

  From Slave_TIMER0 itself: compare 1 event, compare 0 event.

  Interconnection event to Slave_TIMER0: interconnection event 0 (Slave_TIMER1 compare 0 event), interconnection event 2 (Slave_TIMER1 compare 3 event).

  Low-precision events: external event 0(EXEV0), external event 1(EXEV1)

- SHRTIMER_STxCH0RST = 0x0198 0344 and selected events producing “reset request”

are:

From Master_TIMER: compare 0 event, compare 1 event.

From Slave_TIMER0 itself: compare 3 event, period event.

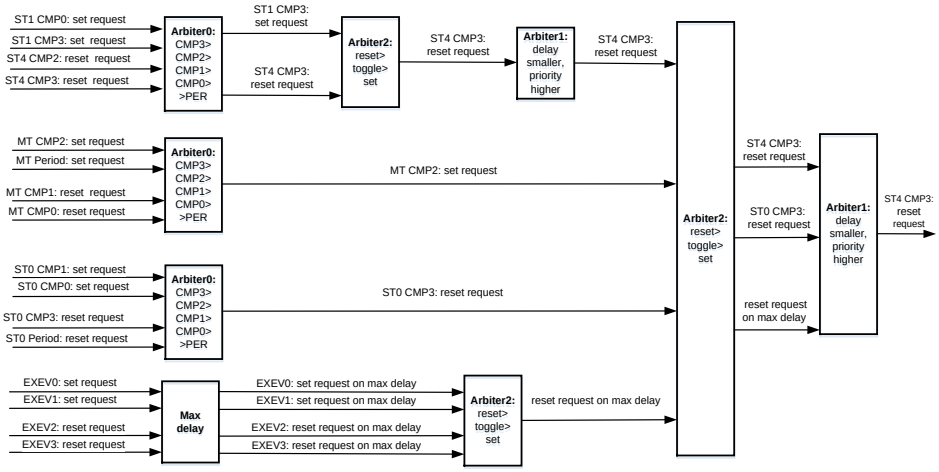
Interconnection event to Slave_TIMER0: interconnection event 7 (Slave_TIMER4 compare 2 event), interconnection event 8 (Slave_TIMER4 compare 3 event).

Low-precision events: external event 2(EXEV2), external event 3(EXEV3)

- The delay: Slave_TIMER4 compare 3 < Slave_TIMER0 compare 3

If the selected events above occur during one t_{HPTMER_CK} period, the arbitration process and results are shown in ***Figure 19-20. Arbitration mechanism example.*** Finally, the "reset request" generated by Slave_TIMER4 compare 3 event is valid during the t_{HPTMER_CK} period and O0PRE will be set to low level.

Figure 19-20. Arbitration mechanism example



Output prepare signal: narrow pulses management

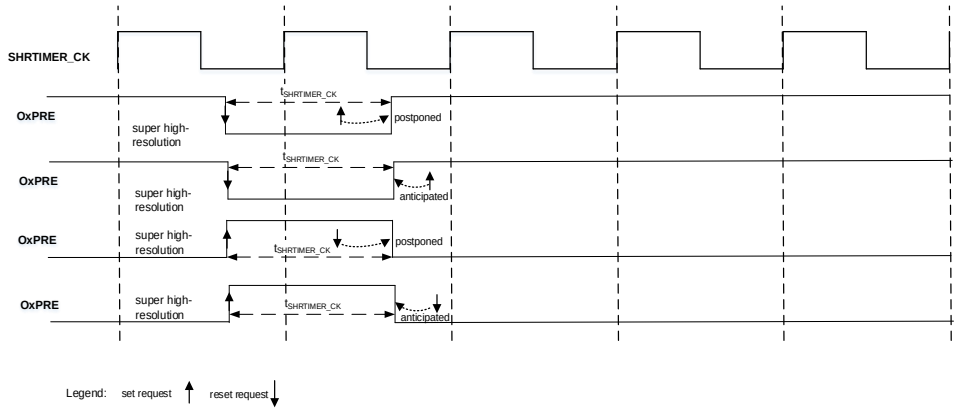
When several "set and/or reset requests" are occurring within 3 consecutive t_{HPTMER_CK} periods, OyPRE(y=0,1) is a narrow pulse. The management of the narrow pulse is different depending on CNTCKDIV[3:0]. It is described in two cases:

- Case 0: CNTCKDIV[3:0] < 4'b0101 or CNTCKDIV[3:0] = 4'b1000
- Case 1: CNTCKDIV[3:0] >= 4'b0101 with CNTCKDIV[3] = 0

Case 0: CNTCKDIV[3:0] < 4'b0101 or CNTCKDIV[3:0] = 4'b1000

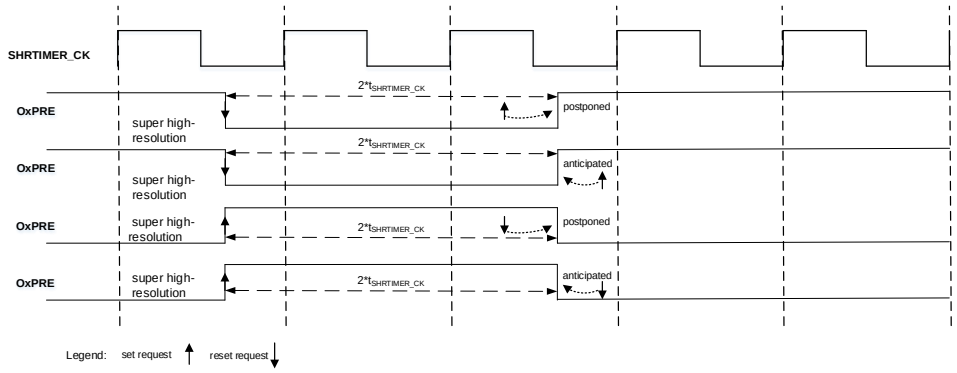
If the "set and reset requests" are generated within two successive t_{HPTMER_CK} period, a pulse of 1 t_{HPTMER_CK} period is generated. Refer to ***Figure 19-21. A pulse of 1 tSHRTMER_CK period***.

Figure 19-21. A pulse of 1 $t_{SHRTMER\_CK}$ period



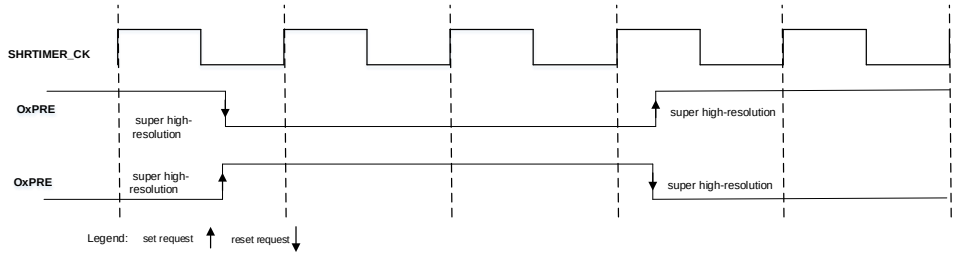
If the “set and reset requests” are generated with an interval including one complete t_{HPTMER_CK} period, a pulse of 2 t_{HPTMER_CK} periods is generated. Refer to ***Figure 19-22. A pulse of 2 tSHRTMER_CK period***.

Figure 19-22. A pulse of 2 $t_{SHRTMER\_CK}$ period



If the “set and reset requests” are generated with an interval including more than two complete t_{HPTMER_CK} periods, the super high-resolution is available. Refer to ***Figure 19-23. Super high-resolution OxPRE wave.***

Figure 19-23. Super high-resolution OxPRE wave



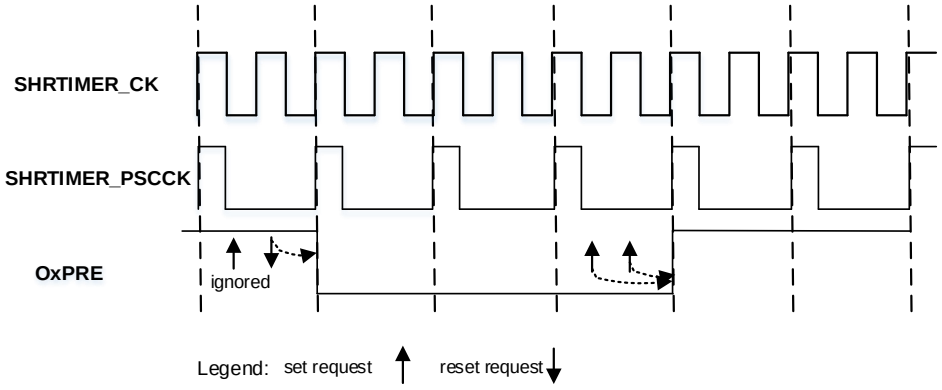
Case 1: CNTCKDIV[3:0] >= 4’b0101 with CNTCKDIV[3] = 0

An “set or reset request” occurring within the SHRTIMER_PSCCK cycle is delayed to the next

active edge of the SHRTIMER_PSCCK, even if the arbitration is still performed every $t_{SHRTIMER\_CK}$ cycle.

When “set and reset requests” from different event sources simultaneously occur in a tSHRTIMER_CK cycle, the “reset request” has the highest priority. In SHRTIMER_PSCCK cycle, subsequent requests override previous requests, and only the last request of that cycle is valid. Refer to ***Figure 19-24. OxPRE wave with CNTCKDIV[3:0] = 4’b0110.***

Figure 19-24. OxPRE wave with CNTCKDIV[3:0] = 4’b0110



Regular mode

When DTEN = 0 in SHRTIMER_STxCHOCTL register and BLNMEN = 0 in SHRTIMER_STxCTL0 register, the set/reset crossbar run in regular mode.

In this mode, C0OPRE and C1OPRE are independent. C0OPRE (C1OPRE) is directly connected to O0PRE(O1PRE).

When the Slave_TIMERx(x=0..4) runs in the RUN or IDEL state and the C0OPRE goes from inactive to active level, CH0OAIF bit in SHRTIMER_STxINTF register will be set to 1, and a output active interrupt or a DMA request is issued if enabled (CH0OAIE = 1 or CH0OADEN = 1 bit in SHRTIMER_STxDMAINTEN register). The CH0OAIF interrupt flag can be cleared by writing 1 to CH0OAIFC bit in SHRTIMER_STxINTFC.

When the Slave_TIMERx(x=0..4) runs in the RUN or IDEL state and the C0OPRE goes from active to inactive level, CH0ONAIF bit in SHRTIMER_STxINTF register will be set to 1, and a output inactive interrupt or a DMA request is issued if enabled (CH0ONAIE = 1 or CH0ONADEN = 1 bit in SHRTIMER_STxDMAINTEN register). The CH0ONAIF interrupt flag can be cleared by writing 1 to CH0ONAIFC bit in SHRTIMER_STxINTFC.

The channel 1 is similar to channel 0.

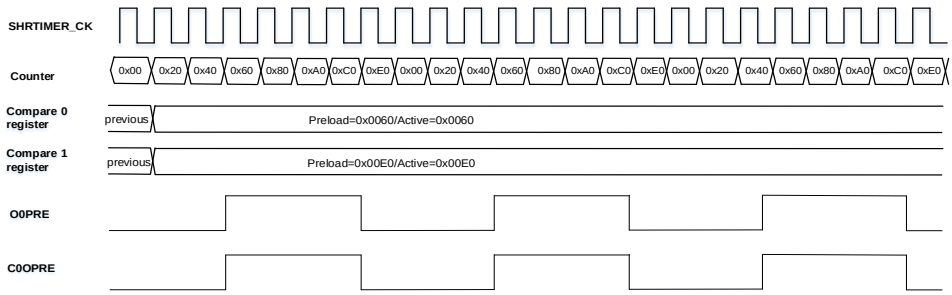
Figure 19-25. C0OPRE wave in regular mode shows C0OPRE wave with the following configuration:

- CNTCKDIV[3:0] = 4’b0000 in SHRTIMER_STxCTL0 register.
- SHRTIMER_STxCH0SET = 0x0000 0008: compare 0 event produces “set request” and

O0PRE will be set to high level.

- SHRTIMER_STxCH0RST = 0x0000 0010: compare 1 event produces “reset request” and O0PRE will be set to low level.
- SHRTIMER_STxCMP0V = 0x0060
- SHRTIMER_STxCMP1V = 0x00E0

Figure 19-25. C0OPRE wave in regular mode



Dead-time mode

When DTEN = 1 in SHRTIMER_STxCHOCTL register and BLNMEN = 0 in SHRTIMER_STxCTL0 register, the set/reset crossbar run in dead-time mode.

In dead-time mode, only O0PRE is programmed to drive C0OPRE and C1OPRE. C0OPRE and C1OPRE are a couple of complementary signals with programmable dead-time insertion between active state transitions.

The dead-time values are defined with DTFCFG[15:0] and DTRCFG[15:0] bit-fields. DTFCFG[15:0] bit-field defines the value of the dead-time following a falling edge of O0PRE, and DTRCFG[15:0] bit-field defines the value of the dead-time following a rising edge of O0PRE.

Note: DTFCFG[8:0] and DTRCFG[8:0] bit-fields are in SHRTIMER_STxDTCTL register. DTFCFG[15:9] and DTRCFG[15:9] bit-fields are in SHRTIMER_STxACTL register

The dead-time values can be positive or negative controlled by DTRS bit and DTFS bit in SHRTIMER_STxDTCTL register. Negative dead-time values can be defined when some waves overlap is required.

The dead-time values are based on a specific clock division according to DTGCKDIV[3:0] bit-field in SHRTIMER_STxDTCTL register.

When the Slave_TIMERx(x=0..4) runs in the RUN or IDEL state and the C0OPRE goes from inactive to active level, CH0OAIF bit in SHRTIMER_STxINTF register will be set to 1, and a output active interrupt or a DMA request is issued if enabled (CH0OAIE = 1 or CH0OADEN = 1 bit in SHRTIMER_STxDMAINTEN register). The CH0OAIF interrupt flag can be cleared by writing 1 to CH0OAIFC bit in SHRTIMER_STxINTFC.

When the Slave_TIMERx(x=0..4) runs in the RUN or IDEL state and the C0OPRE goes from active to inactive level, CH0ONAIF bit in SHRTIMER_STxINTF register will be set to 1, and a output inactive interrupt or a DMA request is issued if enabled (CH0ONAIE = 1 or CH0ONADEN = 1 bit in SHRTIMER_STxDMAINTEN register). The CH0ONAIF interrupt flag can be cleared by writing 1 to CH0ONAIFC bit in SHRTIMER_STxINTFC.

The channel 1 is similar to channel 0.

Figure 19-26. C0OPRE and C1OPRE complementary wave with dead-time shows C0OPRE and C1OPRE wave with O0PRE pulse width greater than the dead-time.

Figure 19-26. C0OPRE and C1OPRE complementary wave with dead-time

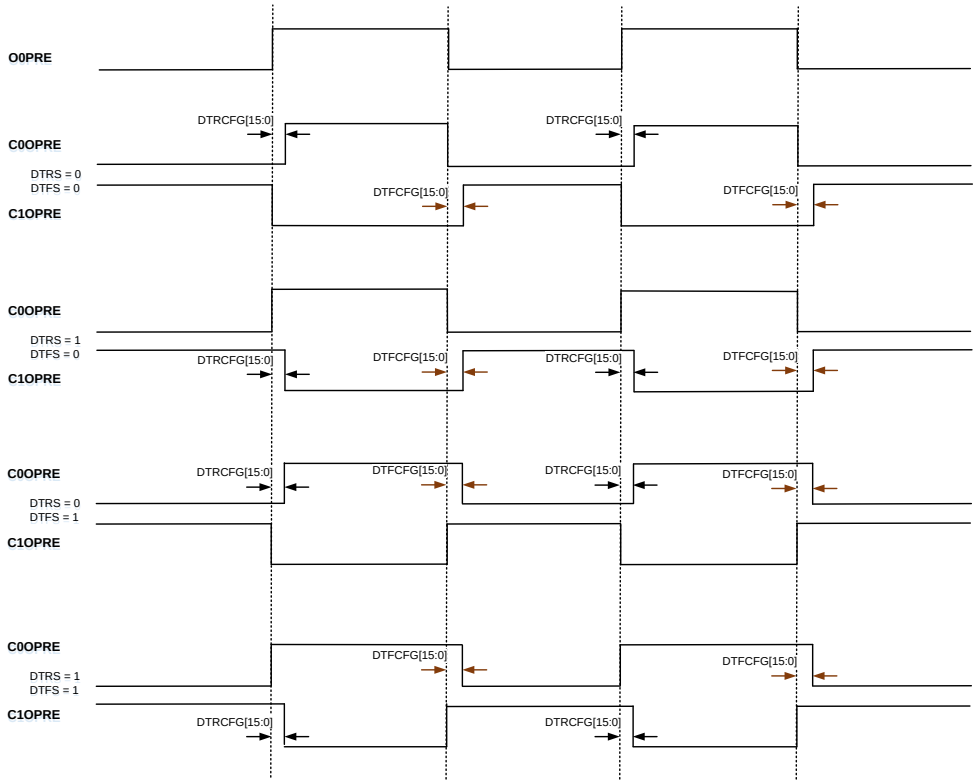
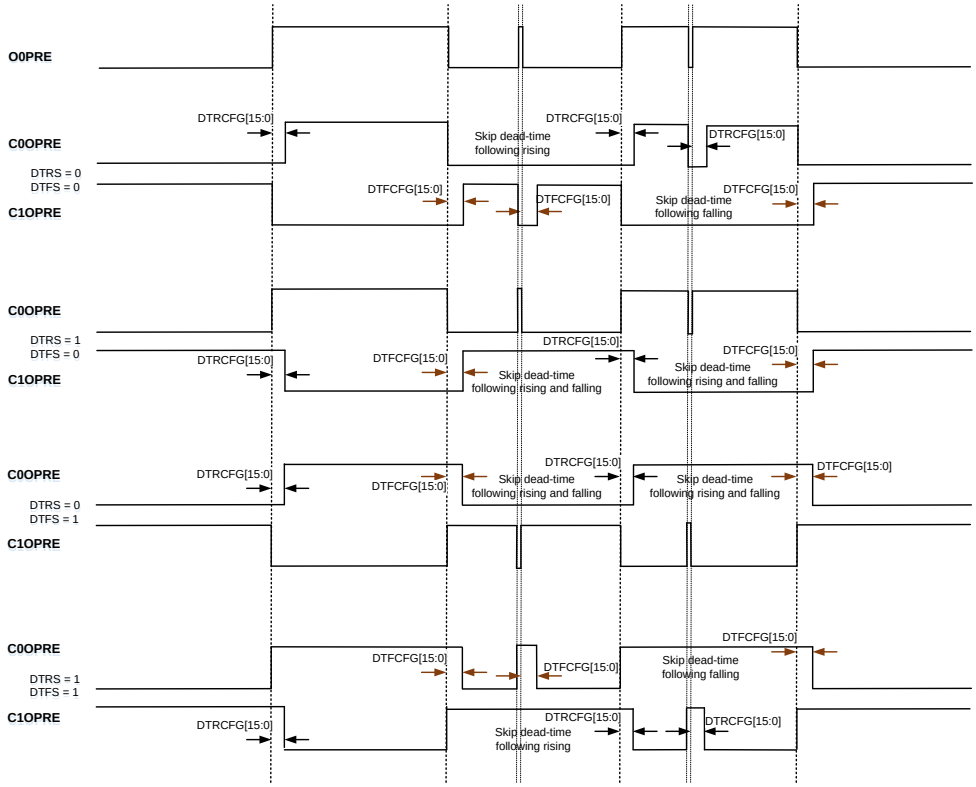


Figure 19-27. Complementary wave with pulse width less than dead-time shows C0OPRE and C1OPRE wave with O0PRE pulse width less than the dead-time.

Figure 19-27. Complementary wave with pulse width less than dead-time

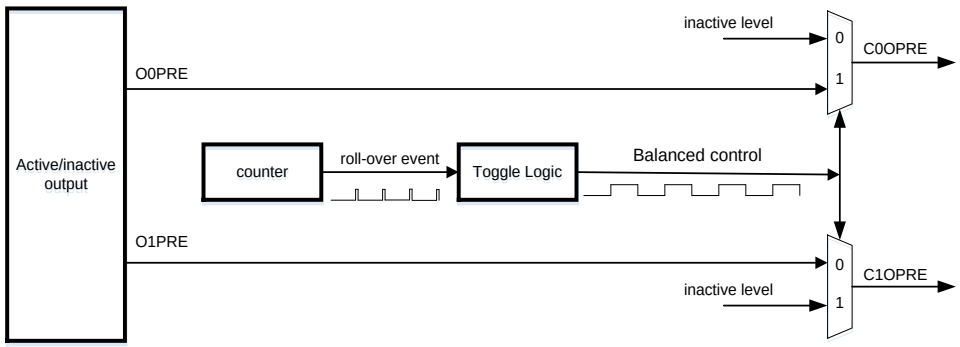


Balanced mode

When DTEN = 0 in SHRTIMER_STxCHOCTL register and BLNMEN = 1 in SHRTIMER_STxCTL0 register, the set/reset crossbar run in balanced mode. The balanced mode is only available when the counter operates in continuous mode and the counter must not be reset once it has been enabled.

Figure 19-28. Structure chart in balanced mode shows the signal control diagram in balanced mode.

Figure 19-28. Structure chart in balanced mode



Once receiving roll-over event, the output of toggle logic module toggles. When the output of toggle logic module is 1(high level), C0OPRE is connected to O0PRE and C1OPRE is set to

inactive level (low level). When the output of toggle logic module is 0(low level), C1OPRE is connected to O1PRE and C0OPRE is set to inactive level (low level).

It is advised to make SHRTIMER_STxCH0SET = SHRTIMER_STxCH1SET and SHRTIMER_STxCH0RST = SHRTIMER_STxCH1RST, in order to achieve a balanced operation with identical waveforms. Still, it is possible to have different programming on both outputs for other uses.

The bit CBLNF in SHRTIMER_STxINTF register which is reset when the balanced mode is disabled, indicates which channel is currently outputting the signal (O0PRE or O1PRE).

When the Slave_TIMERx(x=0..4) runs in the RUN or IDEL state and the C0OPRE goes from inactive to active level, CH0OAIF bit in SHRTIMER_STxINTF register will be set to 1, and a output active interrupt or a DMA request is issued if enabled (CH0OAIE = 1 or CH0OADEN = 1 bit in SHRTIMER_STxDMAINTEN register). The CH0OAIF interrupt flag can be cleared by writing 1 to CH0OAIFC bit in SHRTIMER_STxINTFC.

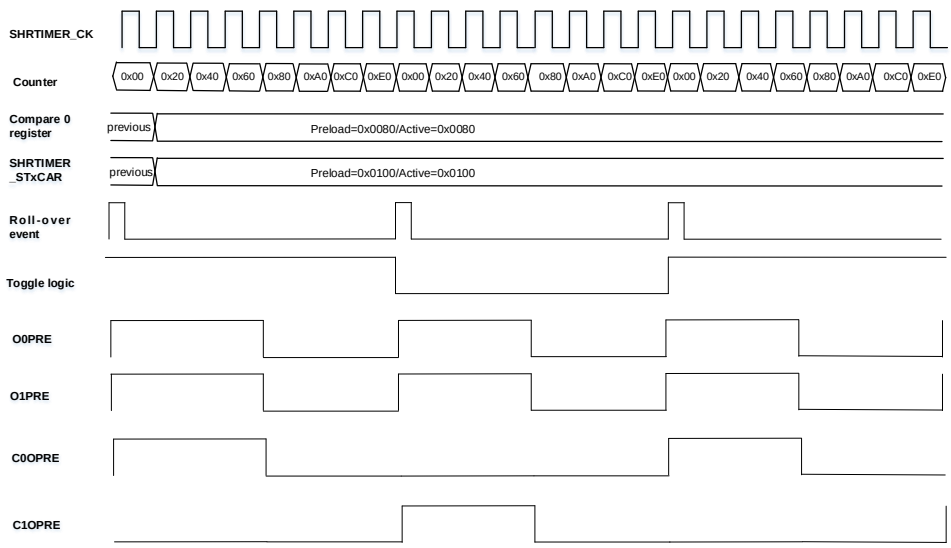
When the Slave_TIMERx(x=0..4) runs in the RUN or IDEL state and the C0OPRE goes from active to inactive level, CH0ONAIF bit in SHRTIMER_STxINTF register will be set to 1, and a output inactive interrupt or a DMA request is issued if enabled (CH0ONAIE = 1 or CH0ONADEN = 1 bit in SHRTIMER_STxDMAINTEN register). The CH0ONAIF interrupt flag can be cleared by writing 1 to CH0ONAIFC bit in SHRTIMER_STxINTFC.

The channel 1 is similar to channel 0.

Figure 19-29. C0OPRE and C1OPRE wave in balanced mode shows C0OPRE and C1OPRE waves with the following configuration:

- SHRTIMER_STxCH0SET = SHRTIMER_STxCH1SET = 0x0000 0004: period event produces “set request”.O0PRE and O1PRE will output high level.
- SHRTIMER_STxCH0RST = SHRTIMER_STxCH1RST = 0x0000 0008: compare 0 event produces “reset request”. O0PRE and O1PRE will output low level.

Figure 19-29. C0OPRE and C1OPRE wave in balanced mode



IDLE control

The stage has three ways to control the IDLE state:

- Delayed IDLE
- Balanced IDLE
- IDLE controlled by bunch mode

Delayed IDLE and balanced IDLE cannot use at the same time. Balanced IDLE is only available in balanced mode. Delayed IDLE or balanced IDLE can use with bunch mode at the same time, but bunch mode has a lowest priority. When set/reset crossbar operates in different modes, different IDLE control operation modes can be used .Refer to ***Table 19-6. Crossbar and IDLE control stage work together.***

Table 19-6. Crossbar and IDLE control stage work together

Set/reset **crossbar operation mode**	**IDLE control stage operation mode**
Regular mode	Delayed IDLE, IDLE controlled by bunch mode
Dead-time mode	Delayed IDLE, Idle controlled by bunch mode
Balanced mode	Either delayed IDLE or balanced IDLE, IDLE controlled by bunch mode

The bits CHyF(y=0,1) in SHRTIMER_STxINTF register indicates the level of CHyOPRE.

Delayed IDLE

When DLYISMEN bit in SHRTIMER_STxCHOCTL register is set to 1, the delayed IDLE is enabled. In delayed IDLE, the “set request” or “reset request” following the selected external event (EXEV5/6 for Slave_TIMER0/1/2 or EXEV7/8 for Slave_TIMER3/4) causes the CHyOPRE (y=0,1) to enter IDLE state. It is associated with ISOy/CHyP(y=0,1) in

SHRTIMER_STxCHOCTL register. Refer to ***Table 19-7. Request to enter in IDLE and exit IDLE state.*** ISOy define the CHyOPRE level in IDLE state. The IDLE mode is permanently maintained but the counter continues to run, until the output is re-enabled to exit delayed IDLE. It is re-enabled by “set request” or “reset request” following overwriting STxCH0EN and STxCH1EN bits to 1.

Table 19-7. Request to enter in IDLE and exit IDLE state

ISOy/CHyP y(=0,1) value	Request to enter in IDLE state	Request exit IDLE state
ISOy = 0, CHyP = 0	“reset request”	“set request” and “reset request”
ISOy = 1, CHyP = 0	“set request”	“set request” and “reset request”
ISOy = 0, CHyP = 1	“set request”	“set request” and “reset request”
ISOy = 1, CHyP = 1	“reset request”	“set request” and “reset request”

The delayed IDLE mode can be applied to a single output (CHyOPRE) or to both outputs (CH0OPRE and CH1OPRE) decided by the bit-field DLYISCH[2:0] in SHRTIMER_STxCHOCTL register, as follows:

- DLYISCH[2:0] = 3’b000: The delayed IDLE mode is applied to CH0OPRE.
- DLYISCH[2:0] = 3’b001: The delayed IDLE mode is applied to CH1OPRE.
- DLYISCH[2:0] = 3’b010: The delayed IDLE mode is applied to CH0OPRE and CH1OPRE.

As soon as the selected external event (EXEV5/6 or EXEV7/8) arrives, the DLYIIF bit in SHRTIMER_STxINTF register is set to 1, and an interrupt or a DMA request is issued if enabled (DLYIIE = 1 or DLYIDEN = 1 bits in SHRTIMER_STxDMAINTEN register). The interrupt flag can be cleared by writing 1 to DLYIIFC bit in SHRTIMER_STxINTFC.

This CHyDLYF(y=0,1) bit in SHRTIMER_STxINTF register indicates the signal CHyOPRE state when the delayed IDLE was triggered by the selected external event (EXEV5/6 or EXEV7/8).

The following four figures show CH0OPRE wave in delayed IDLE with the following configuration:

- C0OPRE is in regular mode: DTEN = 0 in SHRTIMER_STxCHOCTL register and BLNMEN = 0 in SHRTIMER_STxCTL0 register
- Compare 0 event produces “set request”.
- Compare 1 event produces “reset request”.

Figure 19-30. ISO0 = 0 and CHOP = 0 in delayed IDLE

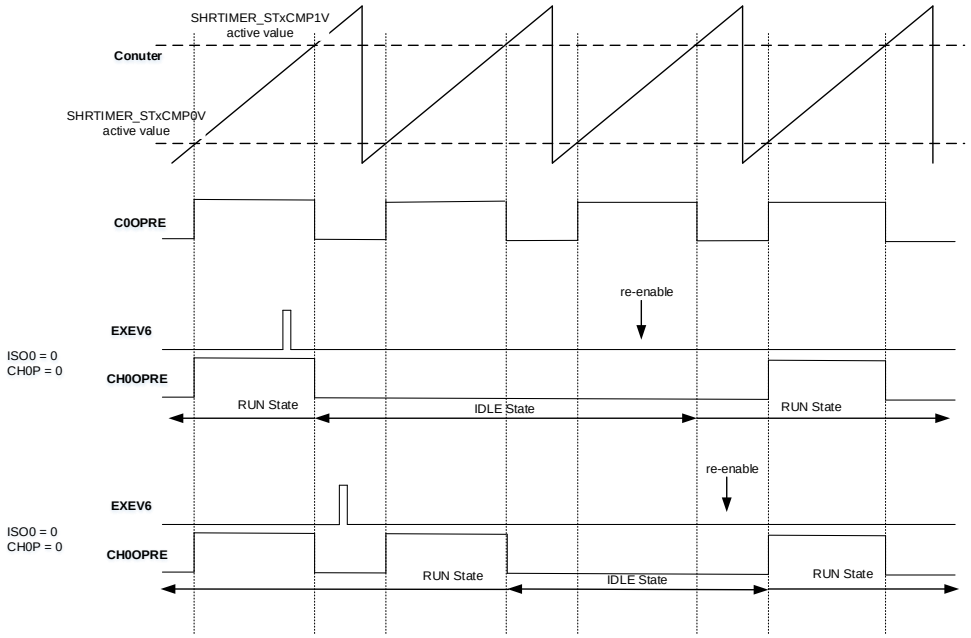


Figure 19-31. ISO0 = 1 and CHOP = 0 in delayed IDLE

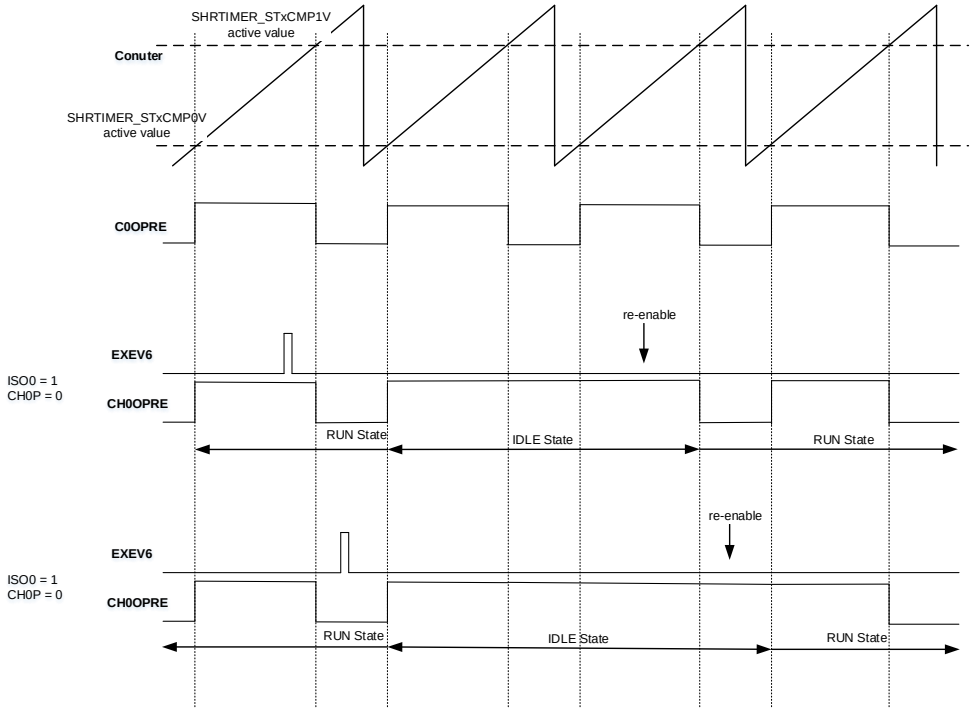


Figure 19-32. ISO0 = 0 and CHOP = 1 in delayed IDLE

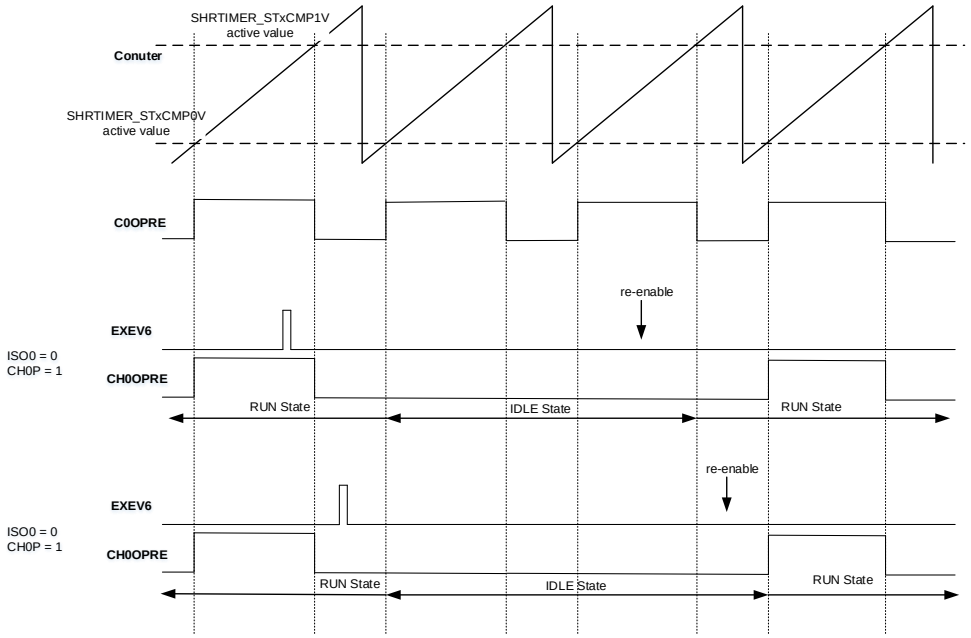
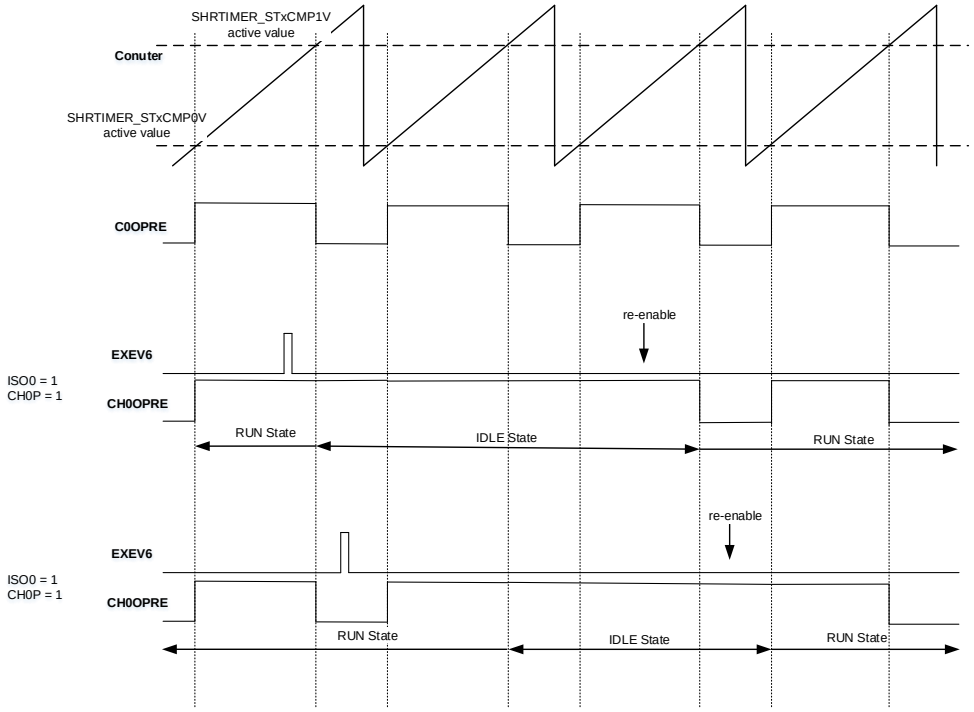


Figure 19-33. ISO0 = 1 and CHOP = 1 in delayed IDLE



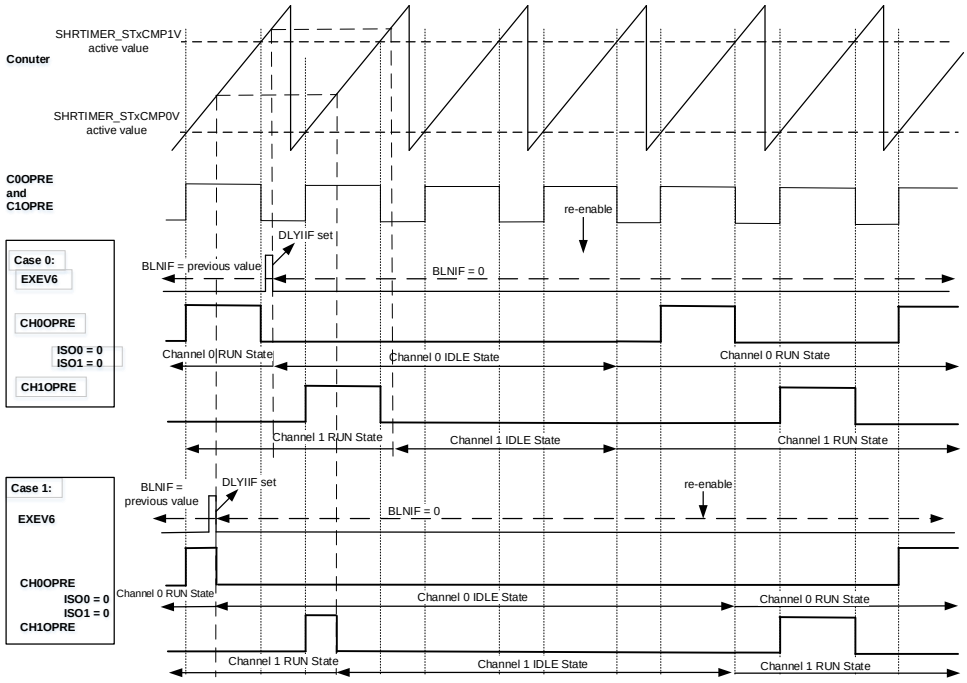
Balanced IDLE

Balanced IDLE is only available in balanced mode. Balanced IDLE is enabled by writing 3'bx11 in DLYISCH[2:0] bit-field in SHRTIMER_STxCHOCTL register. Balanced IDLE is available with external event 5/6 (EXEV5/6) for Slave_TIMER0/1/2, and external event 7/8 (EXEV7/8) for Slave_TIMER3/4.

When the selected event arrives, the CHyOPRE (y=0,1) enters IDLE state and takes the level defined by ISOy bits in the SHRTIMER_STxCHOCTL register. Meanwhile the DLYIIF in SHRTIMER_STxINTF register is set to 1.The selected external event triggers a capture of the counter value into the compare 3 active register (this value is not user-accessible). The balanced mode is maintained for one additional period so that the opposite output CHzOPRE (z=0,1 and z ≠y) can repeat the shorten pulse on CHyOPRE: ***Figure 19-34. Balanced IDEL with ISO0 = 0 and ISO1 = 0*** shows CH0OPRE/ CH1OPRE wave in Slave_TIMER0 in balanced IDLE with the following configuration:

- C0OPRE is in balanced mode: DTEN = 0 in SHRTIMER_STxCHOCTL register and BLNMEN = 1 in SHRTIMER_STxCTL0 register
- Compare 0 event produces “set request”.
- Compare 1 event produces “reset request”.
- Channel 0 and channel 1 output balanced IDLE on external event 6: DLYISCH[2:0] = 111 for Slave_TIMER0

Figure 19-34. Balanced IDEL with ISO0 = 0 and ISO1 = 0



This BLNIF in SHRTIMER_STxINTF register indicates which channel is outputting the signal when a balanced IDLE event entry occurred. For ***Figure 19-34. Balanced IDEL with ISO0 = 0 and ISO1 = 0***, external event 6 (EXEV6) arrives while channel 0 outputs the signal and channel 1 output inactive and BLNIF bit is reset to 0.

The IDLE mode is permanently maintained while the counter continues to run, until the output is re-enabled to exit delayed IDLE. It is re-enabled by “set request” or “reset request” following overwriting STxCH0EN and STxCH1EN bits to 1.

Balanced IDLE can be used together with the bunch mode under the following conditions:

- BMSTx bit must be reset (counter clock(SHRTIMER_PSCCK) is maintained and the counter operates normally)
- No balanced IDLE are triggered while the outputs are in IDLE state controlled by bunch mode.

IDLE controlled by bunch mode

In bunch mode, the IDLE state is controlled by bunch controller. Refer to ***Bunch mode*** for more information.

The balanced IDLE and delayed IDLE has a higher priority than the bunch mode: when the balanced IDLE or delayed IDLE has been triggered, the output is kept with IDLE state after exiting from bunch mode. On the contrary, if the balanced IDLE and delayed IDLE is exited while the bunch mode is active, the bunch mode will be resumed normally.

The bunch mode controller is able to take over the control of any two outputs CHyOPRE (y=0,1). The state of each output during IDLE state controlled by bunch mode is programmed using ISOy and BMCHyIEN(y=0,1) bits in the SHRTIMER_STxCHOCTL register, as in ***Table 19-8. Output during IDEL state controlled by bunch mode***.

Table 19-8. Output during IDEL state controlled by bunch mode

ISOy	BMCHyIEN	CHyOPRE (y=0,1)
x	0	No action: the output is not affected by the bunch controller
0	1	Output inactive during IDLE state controlled by bunch mode
1	1	Output active during IDLE state controlled by bunch mode

Channel output stage

Each Slave_TIMERx unit controls a pair of outputs (STxCH0_O and STxCH1_O). The output stage has three operating states:

- Run state: STxCHy_O(y=0,1) can take the level of CHyOPRE(y=0,1).
- Idle state: STxCHy_O(y=0,1) can take the level defined by ISOy in SHRTIMER_STxCHOCTL register.
- Fault state: STxCHy_O(y=0,1) can be permanently active, inactive or Hi-Z (defined in CHyFLTOS in SHRTIMER_STxCHOCTL register). Refer to ***Fault input*** for more information.

The output stage status is indicated by STxCHyEN bit in SHRTIMER_CHOUTEN register and STxCHyDISF bit in SHRTIMER_CHOUTDISF register, as in ***Table 19-9. Output stage status programming (x=0..4, y=0,1).***

Table 19-9. Output stage status programming (x=0..4, y=0,1)

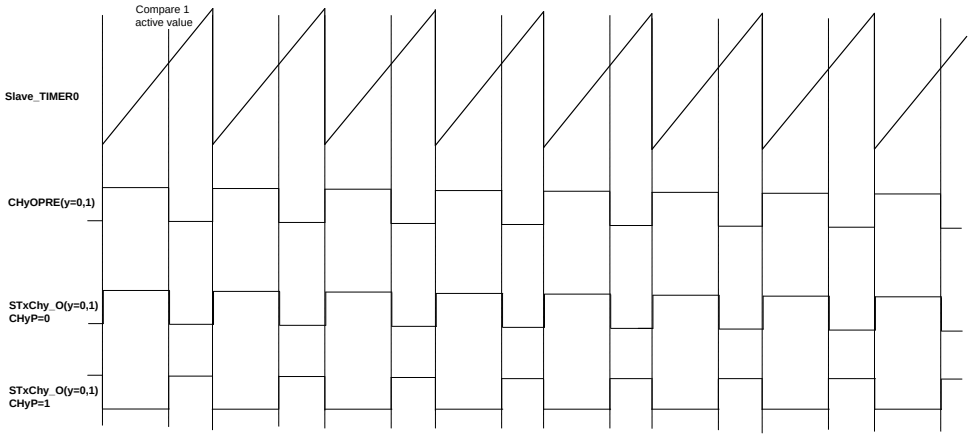
STxCHyEN	STxCHyDISF	output stage status
1	x	Run state
0	0	Idle state
0	1	Fault state

Note: "x" means "0/1".

Writing 1 to STxCHyDIS in SHRTIMER_CHOUTDIS register will disable output and make output stage enters the Idle state. The priority order of the three states is: Idle state > Fault state > Run state.

The output polarity is programmed using CHyP bits in SHRTIMER_STxCHOCTL register. When CHyP = 0, the polarity is output active high. When CHyP = 1, the polarity is output active low. Refer to ***Figure 19-35. STxCHy_O wave with CHyP=0 or CHyP=1.***

Figure 19-35. STxCHy_O wave with CHyP=0 or CHyP=1



The output level in the Fault state is configured using CHyFLTOS[1:0] bits in SHRTIMER_STxCHOCTL register, for each output, as follows:

- 2'b00: output never enters the Fault state and stays in Run or Idle state
- 2'b01: output is active level when in Fault state
- 2'b10: output is inactive level when in Fault state
- 2'b11: output is tri-stated when in Fault state.

The level of the output in Idle state is configured using ISOy bit in SHRTIMER_STxCHOCTL register, as follows:

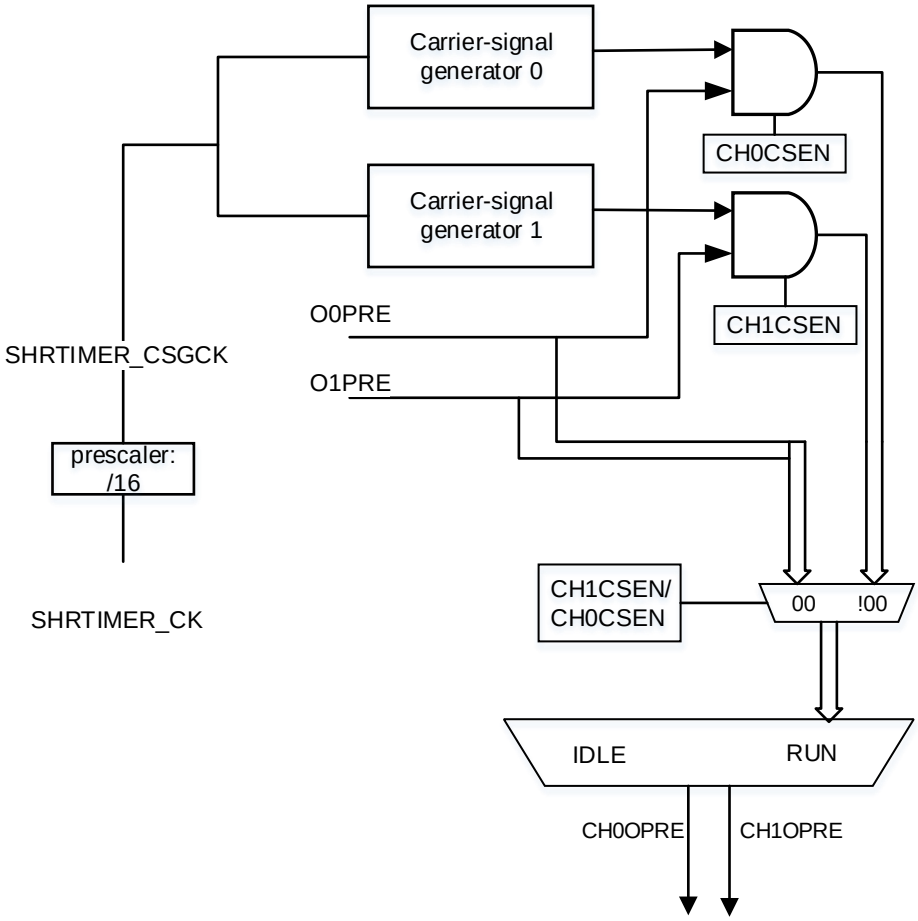
- 2'b0: output is inactive level when in Idle state
- 2'b1: output is active level when in Idle state

Carrier-signal mode

A high-frequency carrier-signal can be added on top of the OyPRE(y=0,1), as follow ***Figure***

19-36. Carrier-signal structure diagram.

Figure 19-36. Carrier-signal structure diagram



In carrier-signal mode, it is possible to define a specific pulse width before the beginning of the carrier-signal. The frequency and duty cycle of the carrier-signal are configurable. Refer to *Figure 19-37. SHRTIMER output with carrier-signal mode enabled.*

Set CH0CSEN and CH1CSEN bits in the SHRTIMER_STxCHOCTL register to 1 to enable carrier-signal mode on channel 0 and 1 respectively.

The pulse width of the first pulse is configured with CSFSTPW[3:0] bit-field in SHRTIMER_STxCSCTL register, following the formula:

$$t_{CSFSTPW}=(CSFSTPW[3:0]+1)* t_{SHRTIMER\_CSGCK},\ \text{where}\ \ t_{SHRTIMER\_CSGCK}=16 *t_{SHRTIMER\_CK}$$

The frequency of carrier-signal is configured with CSPRD[3:0] bit-field in SHRTIMER_STxCSCTL register, following the formula:

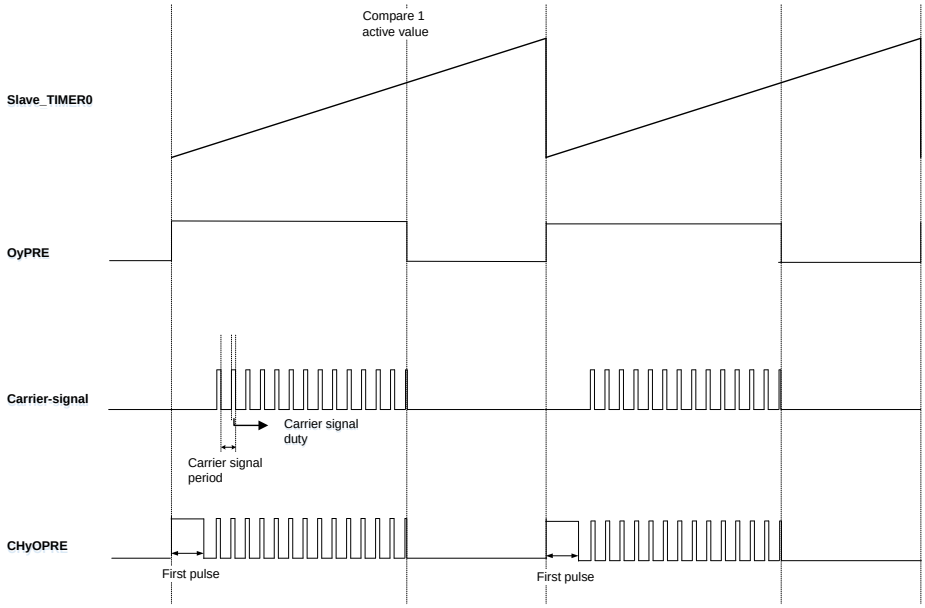
$$t_{CSPRD}=(CSPRD[3:0]+1)* t_{SHRTIMER\_CSGCK},\ \text{where}\ \ t_{SHRTIMER\_CSGCK}=16 *t_{SHRTIMER\_CK}$$

The duty cycle of carrier-signal is configured with CSDTY[2:0] bit-field in SHRTIMER_STxCSCTL register by 12.5% per step.

In carrier-signal mode, the output of carrier-signal generator is combined with the OyPRE

(logic ANDed). The carrier-signal signal is stopped as soon as the OyPRE(y=0,1) is inactive, even if the current carrier period is not completed. Refer to ***Figure 19-37. SHRTIMER output with carrier-signal mode enabled.***

Figure 19-37. SHRTIMER output with carrier-signal mode enabled



Synchronization input start/reset counter

Synchronous input can generate a counter reset event when SYNIRST set 1 in SHRTIMER_STxCTL0 register. Synchronous input can start counter when SYNISTRT set 1 in SHRTIMER_STxCTL0 register. Refer to ***Synchronization input*** for more information.

A synchronization input request will set SYNIIF bit in SHRTIMER_MTINTF register to 1, and a interrupt or a DMA request is issued if enabled(SYNIIE = 1 or SYNIDEN = 1 bits in SHRTIMER_MTDMAINTEN register).The synchronization input interrupt flag can be cleared by writing 1 to SYNIIFC bit in SHRTIMER_MTINTFC.

Update event and shadow registers

Some registers in Slave_TIMERx contain shadow registers. The shadow registers are disabled after MCU reset.

If the SHWEN bit in SHRTIMER_STxCTL0 register is cleared to 0, shadow registers are disabled. The values written into these registers are transferred into active register and take effect immediately.

If the SHWEN bit in SHRTIMER_STxCTL0 register is set to 1, shadow registers are enabled and these registers listed in ***Table 19-10. Slave_TIMERx shadow registers and update event*** are preloaded. The values written into these registers are transferred into the shadow register and do not take effect immediately. Their content of the shadow is transferred into

active register and take effect immediately on update event.

Note: Update event occurs only when SHWEN=1.

Table 19-10. Slave_TIMERx shadow registers and update event lists the registers that contain shadow registers and corresponding update events.

Table 19-10. Slave_TIMERx shadow registers and update event

Registers that contain shadow registers	Shadow registers enable bit	Update event.
SHRTIMER_STxDMAINTEN	SHWEN bit in SHRTIMER_STxCTL0 register	Software(STxSUP bit)
SHRTIMER_STxCAR		Repetition event(UPREP = 1)
SHRTIMER_STxCREP		Counter reset event or roll-over event(UPRST = 1)
SHRTIMER_STxCMP0V		Update event from other timers(UPBSTy for Slave_TIMERy, UPBMT for Master_TIMER)
SHRTIMER_STxCMP0CP		DMA mode end event (UPSEL[1:0] = 4'b0001)
SHRTIMER_STxCMP1V		
SHRTIMER_STxCMP2V		
SHRTIMER_STxCMP3V		Update event following a DMA mode end event (UPSEL[1:0] = 4'b0010)
SHRTIMER_STxDTCTL		
SHRTIMER_STxCH0SET		
SHRTIMER_STxCH0RST		Update event generated on the rising edge of STxUPINy(y=0..2)
SHRTIMER_STxCH1SET		
SHRTIMER_STxCH1RST		Update event following the rising edge of STxUPINy(y=0..2)
SHRTIMER_STxCNTRST		
DTFCFG[15:9] and DTRCFG[15:9] in SHRTIMER_STxACTL		

The update enable inputs STxUPINy(y=0..2) are chip internal signal coming from the general-purpose timers and rising-edge sensitive. Refer to *Table 19-11. STxUPINy(y=0..2) and chip internal signal.*

Table 19-11. STxUPINy(y=0..2) and chip internal signal

Update enable input STxUPINy(y=0..2)	Chip internal signal
STxUPIN0	Reserved
STxUPIN1	Reserved
STxUPIN2	TIM5_TRGO

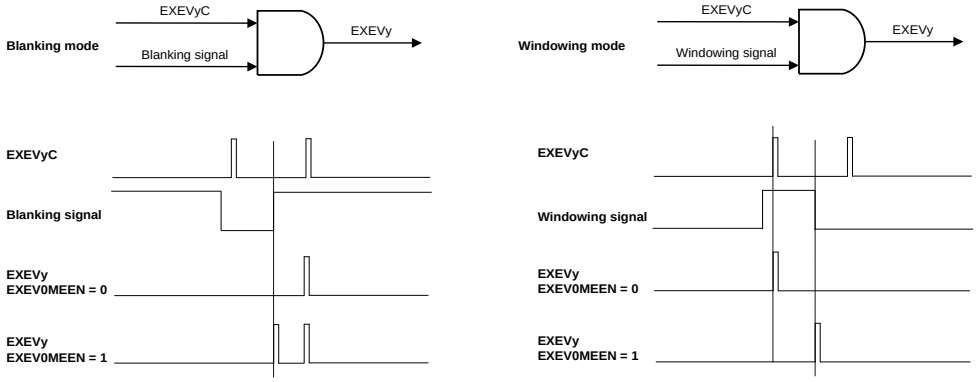
External event filter

The external event EXEVyC(y=0..9) can be filtered to act at a specified time. There are two filter mode:

- Blanking mode: external events that occur within a specified time are ignored.
- Windowing mode: external events that occur within a specified time are taken into account.

Refer to ***Figure 19-38. Blanking mode and windowing mode***

Figure 19-38. Blanking mode and windowing mode



Refer to ***External event*** about the external event EXEVyC(y=0..9) for more information.

Blanking mode

In blanking mode, the external event EXEVyC(y=0..9) that occurs within a specified time is ignored, otherwise it is taken into account. During specified time the blanking signal is low level. This mode is configured with EXEVyFM[3:0] bit-field values ranging from 4'b0001 to 4'b1100.

There are two types of blanking signal sources:

- The Slave_TIMERx itself: the specified time is the duration from the counter reset to the compare event. (EXEVyFM[3:0] = 4'b0001 to 4'b0100 for compare 0 to compare 3 event)
- STBLKSRCz(z=0..7) from other Slave_TIMER units (EXEVyFM[3:0] = 4'b0101 to 4'b1100): the specified time can the duration from the selected Slave_TIMER counter reset to the compare event. It can also be CH1OPRE in the selected Slave_TIMER (in this case, events are ignored as long as the CH1OPRE is low level). Refer to ***Table 19-12. Filtering signals mapping in blanking mode.***

Table 19-12. Filtering signals mapping in blanking mode

To / From	Slave_TIMER 0	Slave_TIMER 1	Slave_TIMER 2	Slave_TIMER 3	Slave_TIMER 4
STBLKSRC0	Slave_TIMER1 compare 0	Slave_TIMER0 compare 0	Slave_TIMER0 compare 1	Slave_TIMER0 compare 0	Slave_TIMER0 compare 1
STBLKSRC1	Slave_TIMER1 compare 3	Slave_TIMER0 compare 3	Slave_TIMER1 compare 0	Slave_TIMER1 compare 1	Slave_TIMER1 compare 0
STBLKSRC2	Slave_TIMER1 CH1OPRE	Slave_TIMER0 CH1OPRE	Slave_TIMER1 compare 3	Slave_TIMER2 compare 0	Slave_TIMER2 compare 0

To / From	Slave_TIMER 0	Slave_TIMER 1	Slave_TIMER 2	Slave_TIMER 3	Slave_TIMER 4
STBLKSRC3	Slave_TIMER2 compare 0	Slave_TIMER2 compare 0	Slave_TIMER1 CH1OPRE	Slave_TIMER2 compare 1	Slave_TIMER2 compare 3
STBLKSRC4	Slave_TIMER2 compare 3	Slave_TIMER2 compare 1	Slave_TIMER3 compare 0	Slave_TIMER2 CH1OPRE	Slave_TIMER2 CH1OPRE
STBLKSRC5	Slave_TIMER2 CH1OPRE	Slave_TIMER2 CH1OPRE	Slave_TIMER3 compare 3	Slave_TIMER4 compare 0	Slave_TIMER3 compare 0
STBLKSRC6	Slave_TIMER3 compare 0	Slave_TIMER3 compare 1	Slave_TIMER3 CH1OPRE	Slave_TIMER4 compare 3	Slave_TIMER3 compare 3
STBLKSRC7	Slave_TIMER4 compare 1	Slave_TIMER4 compare 0	Slave_TIMER4 compare 3	Slave_TIMER4 CH1OPRE	Slave_TIMER3 CH1OPRE

When EXEVyMEEN is set to 1, external event memorized is enabled. The external event is not taken into account immediately. It is memorized and generated as soon as the specified time is completed.

Windowing mode

In windowing mode, the external event EXEVyC(y=0..9) that occurs within a specified time are taken into account, otherwise it is ignored. During specified time the windowing signal is high level. This mode is configured with EXEVyFM[3:0] bit-field values ranging from 4'b1101 to 4'b1111.

If no EXEVyC(y=0..9) occurs within a specified time, the timeout event will be generated at the end of the specified time.

There are two types of windowing signal sources:

- The Slave_TIMERx itself: the specified time is the duration from the counter reset to the compare event. (EXEVyFM[3:0] = 4'b1101 and 4'b1110 ,respectively compare 1 and compare 2).
- STWDSRC from other Slave_TIMER units (EXEVyFM[3:0] = 4'b1111): the specified time is the duration from the selected Slave_TIMER counter reset to compare event. Refer to ***Table 19-13. Filtering signals mapping in windowing mode.***

Table 19-13. Filtering signals mapping in windowing mode

To / From	Slave_TIMER0	Slave_TIMER1	Slave_TIMER2	Slave_TIMER3	Slave_TIMER4
STWDSRC	Slave_TIMER1 compare 1	Slave_TIMER0 compare 1	Slave_TIMER3 compare 1	Slave_TIMER2 compare 1	Slave_TIMER3 compare 1

When EXEVyMEEN is set to 1, external event memorized is enabled. The external event is not taken into account immediately. It is memorized and generated as soon as the specified time is completed.

DAC trigger

When the Slave_TIMERx update event occurs with DACTRGS[1:0] != 2'b00 in SHRTIMER_STxCTL0 register, a DAC trigger request can be generated on SHRTIMER_DACTRIGOy(y=0..3). If DACTRGS[1:0] = 2'b00 in SHRTIMER_STxCTL0 register, Slave_TIMERx won't generate DAC trigger request. SHRTIMER_DACTRIGOy(y=0..3) is the internal signal connected from Slave_TIMERx to the DAC module. Refer to ***Trigger to DAC*** for more information.

19.4.3. DLL calibrate

The DLL can produce and calibrate a super high resolution clock SHRTIMER_HPCK ($f_{SHRTIMER\_HPCK}$ = 64 *$f_{SHRTIMER\_CK}$). DLL can calibrate the super high resolution clock SHRTIMER_HPCK either once or periodically.

When CLBPEREN bit in SHRTIMER_DLLCCTL register is set to 1, it enables the periodic DLL calibration and the bit-field CLBPER[1:0] decides the calibration period. The DLL will periodically calibrate the clock during the entire SHRTIMER running.

When CLBPEREN bit in SHRTIMER_DLLCCTL register is cleared to 0, DLL calibrates the super high resolution clock SHRTIMER_HPCK only once by writing CLBSTRT to 1.

19.4.4. Bunch mode

The bunch mode controller allows to have the CHyOPRE (y=0,1) alternatively in IDLE and RUN state by hardware. This mode is enabled with BMEN bit in the SHRTIMER_BMCTL register and usually used in light load situations.

The bunch mode controller consists of:

- A counter called BM-counter.
- A compare register: SHRTIMER_BMCMPV, to define IDLE duration.
- A period register: SHRTIMER_BMCAR, to define the sum of the IDLE and RUN duration.

BM-counter counting mode

BM-counter can operate in continuous or single pulse mode.

When BMCTN = 1, BM-counter operate in continuous mode. The BM-counter counts up continuously from 0 to the counter-reload value(defined in SHRTIMER_BMCAR register). When counts up to the counter-reload value (SHRTIMER_BMCAR), the counter restarts from 0. The bunch mode operation is maintained until BMOPTF bit in SHRTIMER_BMCTL is reset to terminate it.

When BMCTN = 0, BM-counter operate in single pulse mode. The BM-counter counts up continuously from 0 to the counter-reload value (SHRTIMER_BMCAR). When counts up to

the counter-reload value (SHRTIMER_BMCAR), the BM-counter is stopped.

When counts up to the counter-reload value (SHRTIMER_BMCAR), BMPERIF in SHRTIMER_INTF register is set to 1 and bunch mode controller generates a bunch mode period interrupt request if BMPERIE=1 in SHRTIMER_INTEN register. The BMPERIF bit can be cleared by writing 1 to BMPERIFC bit in SHRTIMER_INTC register.

Bunch mode timing

The BM-counter can be clocked by several sources, selected with BMCLKS[3:0] bits in the SHRTIMER_BMCTL register. When the rising edge of the selected clock source signal arrives, BM-counter increments by 1.

When BMCLKS[3:0]=4'b1010, the clock source of BM-counter is the $f_{SHRTIMER\_CK}$ prescaled by a factor defined with BMPSC[3:0] bit-field in SHRTIMER_BMCTL register.

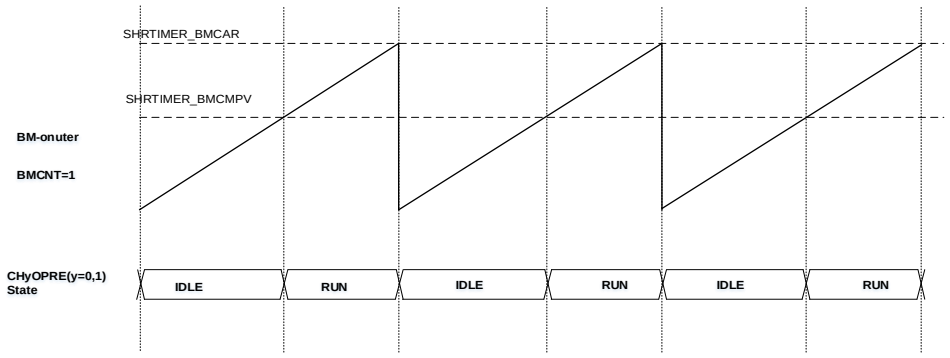
When BMCLKS[3:0]= 4'b0110 to 1001, the clock source of BM-counter is chip internal signal : BMCLKy(y=0..3). Refer to ***Table 19-14. Chip internal signal in bunch mode.***

Table 19-14. Chip internal signal in bunch mode

BMCLKy(y=0..3)	Chip internal signal
BMCLK0	Reserved
BMCLK1	Reserved
BMCLK2	TIMER6_TRGO
BMCLK3	Reserved

The duration of the IDLE is defined with SHRTIMER_BMCMPV register, and SHRTIMER_BMCAR register defines the bunch mode period which is the sum of the IDLE and RUN duration.Refer to ***Figure 19-39. Bunch mode timing chart***.

Figure 19-39. Bunch mode timing chart



When BMSE is set，the SHRTIMER_BMCMPV and SHRTIMER_BMCAR registers are preloaded and the transfer from preload to active register happens:

- when the bunch mode is enabled (BMEN = 1)
- at the end of the bunch mode period

Notice that a write into the SHRTIMER_BMCAR register disables the update temporarily, until

the SHRTIMER_BMCMPV compare register is written.

Bunch mode entry

There are 32 events defined in SHRTIMER_BMSTRG register can be used to trigger the bunch mode operation. These trigger events can be selected simultaneously and are logic ORed. During BM-counter counting process, these trigger events are ignored. These trigger events are divided into seven categories:

1. Events from Master_TIMER: repetition event, reset/roll-over event, compare 0 to 3 event
2. Events from Slave_TIMERx: repetition event, reset/roll-over event, compare 0 and 1 event
3. External event: EXEV6 and EXEV7
4. Slave_TIMER0 period event following EXEV6
5. Slave_TIMER3 period event following EXEV7
6. Chip internal signal: TIMER6_TRGO
7. Software: Writing 1 to the SWTRG bit in SHRTIMER_BMSTRG register.

When the trigger event occurs, there are two ways to enter bunch mode: regular entry and delayed entry.

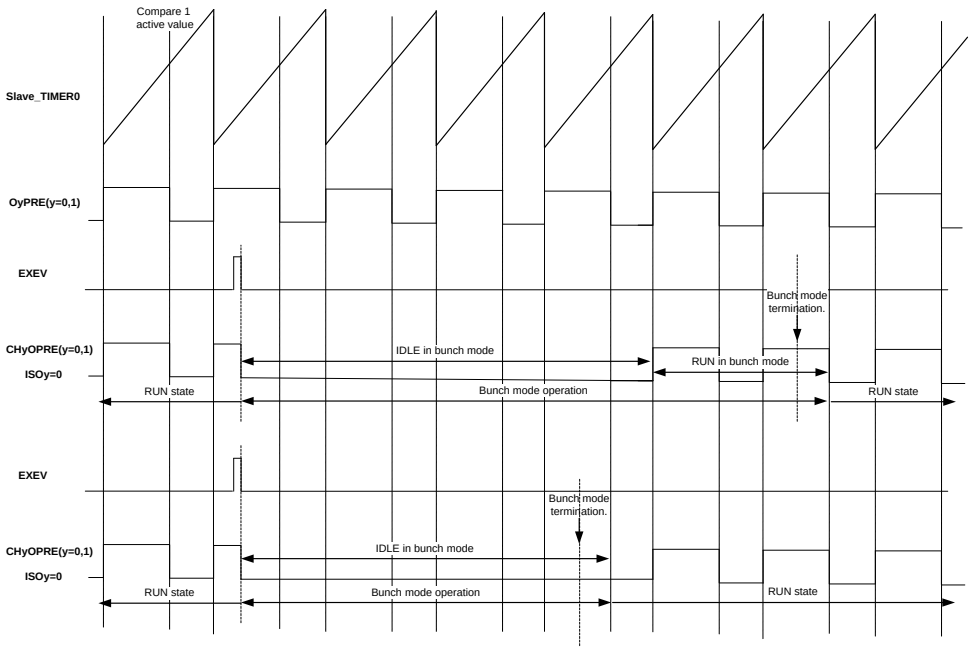
Regular entry

When BMCHyDTI(y=0,1) bit in SHRTIMER_STxCHOCTL register is 0, the bunch mode entry is regular. The output will enter the bunch mode and take their idle level (as per ISO0 and ISO1 setting) on the first BM-counter counting clock after the selected event occurs.

Figure 19-40. Regular entry for bunch mode shows CHyOPRE wave in Slave_TIMER0 with the following configuration:

- CyOPRE is in regular mode: DTEN = 0 in SHRTIMER_ST0CHOCTL register and BLNMEN = 0 in SHRTIMER_ST0CTL0 register
- Period event produces “set request”.
- Compare 1 event produces “reset request”.
- Clock source of BM-counter is roll-over event in Slave_TIMER0: BMCLKS[3:0]=4’b0001

Figure 19-40. Regular entry for bunch mode



Delayed entry

When BMCHyDTI(y=0,1) bit in SHRTIMER_STxCHOCTL register is 1, the bunch mode entry is delayed. CHyOPRE forces to a dead-time insertion before entering bunch mode.

Each CHyOPRE has its own dead-time value:

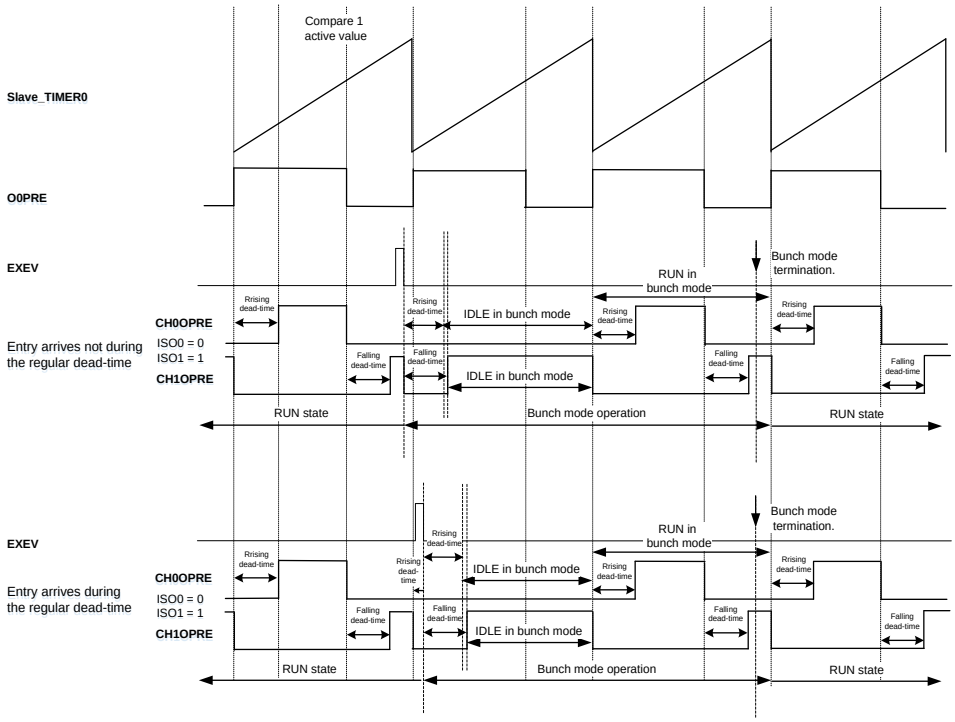
- DTRCFG[15:0] on CH0OPRE when BMCH0DTI=1
- DTFCFG[15:0] on CH1OPRE when BMCH1DTI=1

Delayed entry applies to the situation that one of the CHyOPRE has an active IDLE level (ISOy = 1) and dead-times are positive (DTRS/DTFS set to 0).

When the bunch mode entry arrives during the regular dead-time, it is aborted and a new dead-time is re-started corresponding to the inactive period. Refer to ***Figure 19-41. Delayed entry for bunch mode*** with the following configuration:

- CyOPRE is in dead-time mode: DTEN = 1 in SHRTIMER_ST0CHOCTL register and BLNMEN = 0 in SHRTIMER_ST0CTL0 register
- Period event produces “set request”.
- Compare 1 event produces “reset request”.
- Clock source of BM-counter is roll-over event in Slave_TIMER0: BMCLKS[3:0]=4’b0001

Figure 19-41. Delayed entry for bunch mode



Bunch mode exit

In continuous mode, the bunch mode exit is forced by software. The bunch mode exit by “set request” or “reset request” following overwriting BMOPTF or BMEN bit to 0. Refer to ***Figure 19-40. Regular entry for bunch mode*** and ***Figure 19-41. Delayed entry for bunch mode***.

In single pulse mode, the bunch mode exit once the IDLE period is elapsed.

Counter clock in bunch mode

The counters of the Master_TIMER and Slave_TIMERx(x=0..4) units can be stopped and reset during the bunch operation (RUN+IDLE).The bits BMMT and BMSTx(x=0..4) in SHRTIMER_BMCTL register is used for this purpose:

- BMMT or BMSTx(x=0..4) equal to 0: Master_TIMER or Slave_TIMERx(x=0..4) counter clock(SHRTIMER_PSCCK) is maintained and the counter operates normally
- BMMT or BMSTx(x=0..4) equal to 1: Master_TIMER or Slave_TIMERx(x=0..4) counter clock(SHRTIMER_PSCCK) is stopped and the counter is reset.

Use SHRTIMER_STxCMP0CP register to emulate bunch mode

The SHRTIMER_STxCMP0CP register can be used to produce a waveform similar to that controlled by bunch mode. To do this, the following configuration is required:

- Compare 0 event is used to produce “reset request”.

- Period event is used to produce “set request”.
- Write two 32-bit data to the SHRTIMER_STxCMP0CP register consecutively using DMA (upon repetition event), as below:

  SHRTIMER_STxCMP0CP = {CREP[7:0] = (RUN number of periods - 1); CMP0VAL[15:0] = duty cycle}

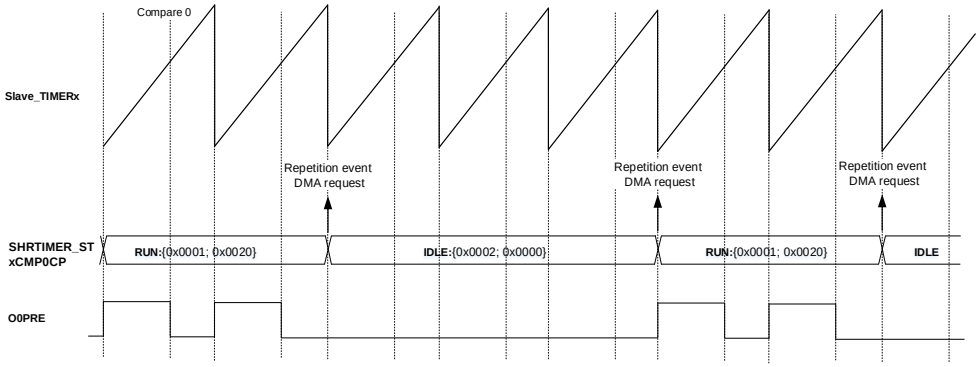
  SHRTIMER_STxCMP0CP = {CREP[7:0] = (IDLE number of periods - 1); CMP0VAL[15:0] = 0}

For example, to generate a PWM wave with 2 periods active every 5 PWM periods, the following values can be used:

- RUN: SHRTIMER_STxCMP0CP = {0x0001; 0x0020}
- IDLE: SHRTIMER_STxCMP0CP = {0x0002; 0x0000}

Its behavior is shown ***Figure 19-42. Emulate bunch mode example.***

Figure 19-42. Emulate bunch mode example



19.4.5. Synchronization input/output

The synchronization circuitry is controlled inside the Master_TIMER:

- Synchronization output: the SHRTIMER can generate a synchronization signal as a master.
- Synchronization input: SHRTIMER can also wait for a trigger to be synchronized as a slaver.

Synchronization output

SHRTIMER can be configured to synchronize external resources and act as a master unit.

The bit-field SYNOSRC[1:0] in SHRTIMER_MTCTL0 register can be configured to selecte the source to be sent to the synchronization output. There are four sources available:

- 2’b00: Master_TIMER start event. There are three situations in which the generated start event can be used as synchronous output: when MTCEN bit is set to 1, when the timer

is re-started after having reached the period value in single pulse mode, a reset which occurs during the counting with CTNM or CNTRSTM bits set to 1.

- 2'b01: Master_TIMER compare 0 event
- 2'b10: Slave_TIMER0 reset and start event. It is similar to Master_TIMER start event, except for the following: counter roll-over in continuous mode, discarded reset request in single pulse mode with CNTRSTM=0.
- 2'b11: Slave_TIMER0 compare 0 event

The bit-field SYNOPLS[1:0] in SHRTIMER_MTCTL0 register specifies the polarity of the synchronization output signal:

- 2'b00: Pulse generated disable. No pulse on the synchronization output pad SHRTIMER_SCOUT.
- 2'b01: Reserved.
- 2'b10: Positive pulse generated on the synchronization output pad SHRTIMER_SCOUT. The length of the positive pulse is 16 $t_{SHRTIMER\_CK}$ cycles.
- 2'b11: Negative pulse generated on the synchronization output pad SHRTIMER_SCOUT. The length of the negative pulse is 16 $t_{SHRTIMER\_CK}$ cycles.

Synchronization input

SHRTIMER can wait for a trigger to be synchronized as a slaver. The bit-field SYNISRC[1:0] in SHRTIMER_MTCTL0 register can be configured to select the synchronization input source. There are four trigger sources available:

- 2'b00: Synchronization input disable.
- 2'b01: Reserved.
- 2'b10: Chip internal signal. TIMER0_TRGO in the advanced timer TIMER0
- 2'b11: Chip external pin. A positive pulse on the chip external pin (SHRTIMER_SCIN) is effective (rising-edge sensitive).

The Master_TIMER behavior is defined with the bits SYNISTRT and SYNIRST in SHRTIMER_MTCTL0 register. The Slave_TIMERx behavior is defined with the bits SYNISTRT and SYNIRST in SHRTIMER_STxCTL0 register.

When SYNISTRT is set to 1, the synchronization input signal starts the timer's counter provided that the counter must first enable by setting STxCEN or MTCEN bit to 1. In continuous mode, the counter will not start even if STxCEN or MTCEN bit is set to 1, but only after the synchronous input signal arrives.

When SYNIRST is set to 1, the synchronization input signal resets the counter and decrements the repetition counter as any other reset event.

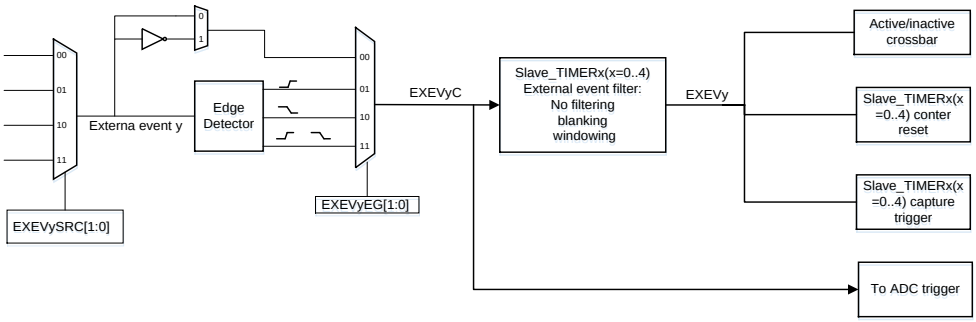
19.4.6. External event

There are 10 external events that can be simultaneously used on five Slave_TIMER. The external event y(y=0..4) are configured using the SHRTIMER_EXEVCFG0 register , and the

external event y(y=5..9) are configured using the SHRTIMER_EXEVCFG0 and SHRTIMER_EXEVDFCTL registers.

The process by which external event y(y=0..4) are processed is shown in ***Figure 19-43. Extern event y(y=0..4) processed diagram***.

Figure 19-43. Extern event y(y=0..4) processed diagram

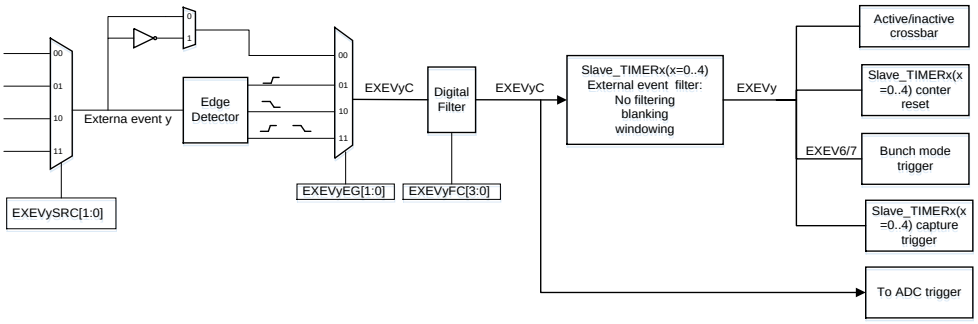


The configuration of external event y(y=0..4) is as follows:

- Up to 4 sources: configured with the EXEVySRC[1:0] bit-field.
- The sensitivity selection: configured with EXEVyEG[1:0] bit-field. It can be either level sensitive or edge sensitive (rising, falling or both).
- The polarity selection: configured with EXEV0P bit in case of a level sensitivity (EXEVyEG[1:0]=2'b00).

The process by which external event y(y=5..9) are processed is shown in ***Figure 19-44. Extern event y(y=5..9) processed diagram***.

Figure 19-44. Extern event y(y=5..9) processed diagram



The configuration of external event y(y=5..9) is as follows:

- Up to 4 sources: configured with the EXEVySRC[1:0] bit-field.
- The sensitivity selection: configured with EXEVyEG[1:0] bit-field. It can be either level sensitive or edge sensitive (rising, falling or both).
- The polarity selection: configured with EXEV0P bit in case of a level sensitivity (EXEVyEG[1:0]=2'b00).
- Digital filters configuration: configured with EXEVyFC[3:0] bit-field in the SHRTIMER_EXEVDFCTL register.

The digital filters sampling clock $f_{SHRTIMER\_EXEVFCK}$ is defined with EXEVFDIV[2:0] bit-field in SHRTIMER_EXEVDFCTL register.

These external events sources EXEVySRCz(y=0..9,z=0..4) can come from comparators, digital input pins, ADC's analog watchdogs and TIMER_TRGO. Refer to ***Table 19-15. External events mapping.***

Table 19-15. External events mapping

External event	EXEVySRC0	EXEVySRC1	EXEVySRC2	EXEVySRC3
External event 0	PC12	Comparator 1	TIMER0_TRGO	ADC0_AWD0
External event 1	PC11	Comparator 3	TIMER1_TRGO	ADC0_AWD1
External event 2	PB7/PD5	Comparator 5	TIMER2_TRGO	ADC0_AWD2
External event 3	PB6/PG11	x	x	ADC1_AWD0
External event 4	PB9/PG12	x	x	ADC1_AWD1
External event 5	PB5	Comparator 1	TIMER5_TRGO	ADC1_AWD2
External event 6	PB4	Comparator 3	TIMER6_TRGO	x
External event 7	PB8	Comparator 5	x	x
External event 8	PB3	x	x	x
External event 9	PC6/PG13	x	x	x

Note: "×" means not available.

The extern event y(y=0..9) can be used and also be filtered to have an action limited in a specified time. Refer to ***External event filter.***

19.4.7. Fault input

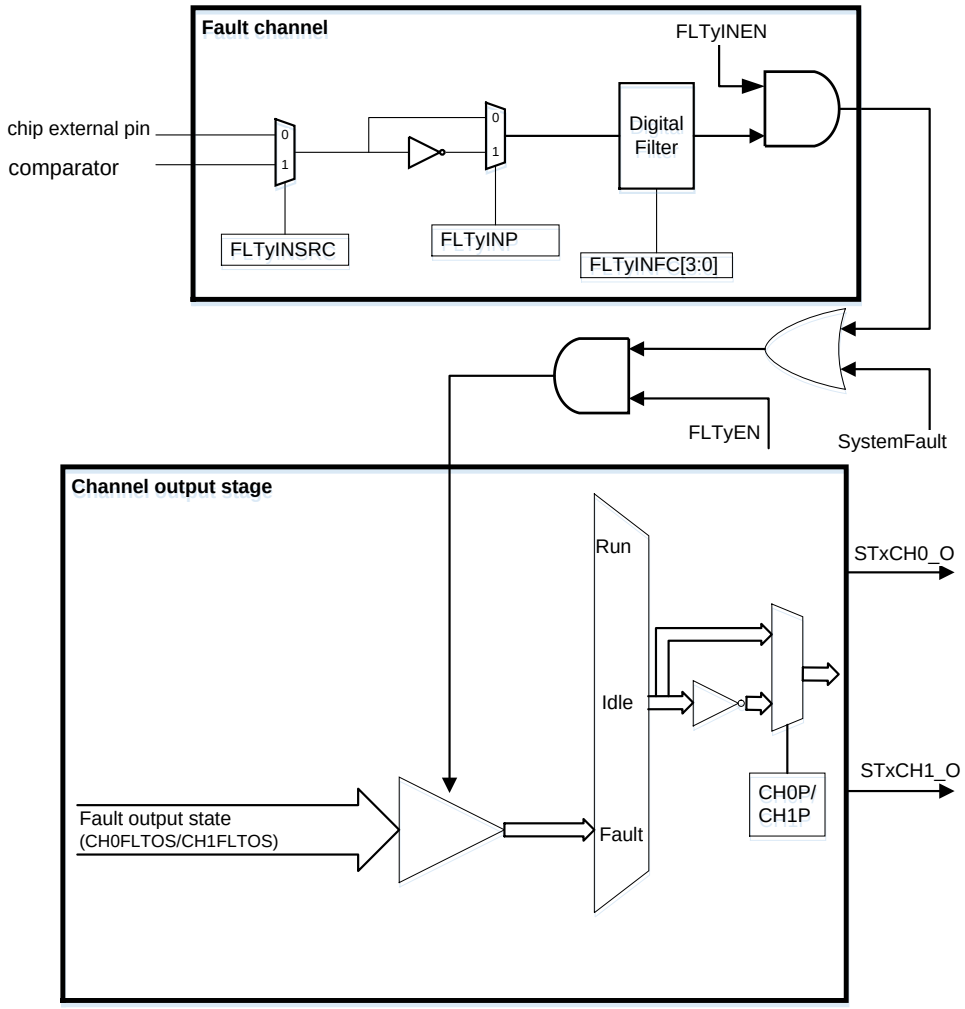
SHRTIMER has a fault protection mechanism which is available for each Slave_TIMERx. Refer to ***Figure 19-45. Fault input diagram.***

When a fault event occurs, the output (STxCHy_O, x=0..4,y=0,1)is at a predefined level, which is maintained until the output is re-enabled by software (writing 1 to STxCHyEN bit).

The predefined level is configured by bit-field CHyFLTOS[1:0] in SHRTIMER_STxCHOCTL register. The protection mechanism can handle two types of fault sources:

- Fault channel: fault event from digital input pin or comparator.
- System fault: signals coming from inside the MCU, for example the Cortex®-M33-lockup signal.

Figure 19-45. Fault input diagram



Fault input is enabled using FLTyEN bit in SHRTIMER_STxFLTCTL register. A write-once FLTENPROT bit in the SHRTIMER_STxFLTCTL register allows to protect FLTyEN bits for safety purpose. When FLTENPROT bit is set to 1, FLTyEN bits are writing protected (read-only).

Fault channel

The fault channel is fully configurable using SHRTIMER_FLTINCFG0 and SHRTIMER_FLTINCFG1 registers.

FLTyINSRC (y=0..4) bit selects the source of the fault channel, that can be from either a digital input pin or comparator output. Refer to ***Table 19-16. Fault channel mapping***

Table 19-16. Fault channel mapping

Fault channel	FLTyINSRC = 0 (input pin)	FLTyINSRC = 1(comparator)
Fault channel 0	PA12	Comparator 1
Fault channel 1	PA15	Comparator 3
Fault channel 2	PB10/PD4	Comparator 5
Fault channel 3	PB11	x

Fault channel	FLTyINSRC = 0 (input pin)	FLTyINSRC = 1(comparator)
Fault channel 4	PC7/PG10	x

Note: "x" means not available.

The polarity of the signal can be configured by the FLTyINP polarity bit in SHRTIMER_FLTINCFG0 and SHRTIMER_FLTINCFG1 registers. If FLTyINP = 0, the signal is active at low level; if FLTyINP = 1, it is active when high.

The digital filters of the signal after the polarity setting can be configured by the FLTyINFC[3:0] bit-field in SHRTIMER_FLTINCFG0 and SHRTIMER_FLTINCFG1 registers. The digital filters sampling clock $f_{SHRTIMER\_FLTFCK}$ is defined with FLTFDIV[2:0] bit-field in SHRTIMER_FLTINCFG1 register.

The fault channel y(y=0..4) can be enabled using bits FLTyINEN in SHRTIMER_FLTINCFG0 and SHRTIMER_FLTINCFG1 registers, and they can be selected simultaneously.

A write-once FLTyINPROT bit in SHRTIMER_FLTINCFG0 and SHRTIMER_FLTINCFG1 registers allows to protect FLT0INEN, FLT0INP, FLT0INSRC and FLT0INFC[3:0] bits for safety purpose. When FLTyINPROT bit is set to 1, these bits are writing protected (read-only).

System fault

The system fault comes from the signal inside the chip:

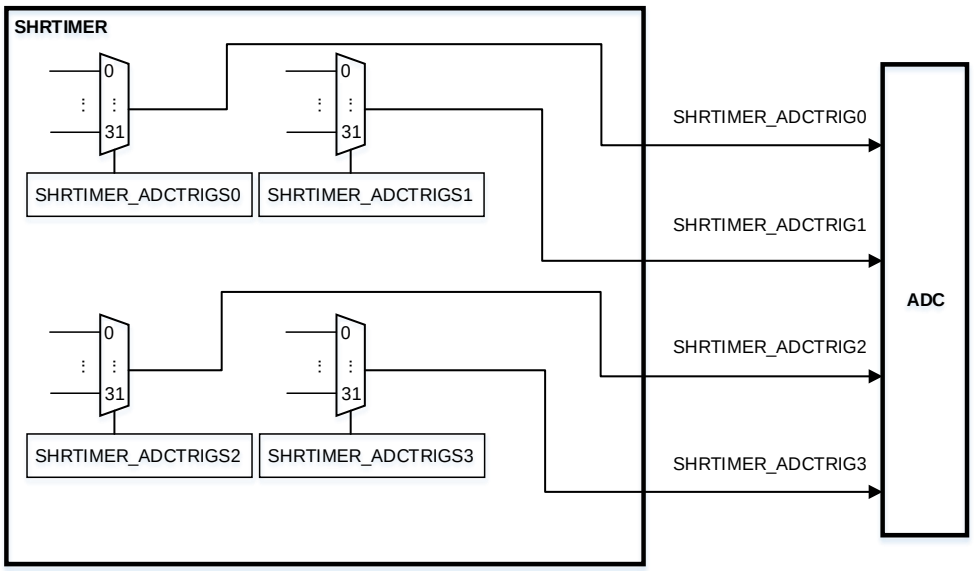
- HXTAL failure event generated by Clock Monitor.
- The Cortex®-M33-lockup signal.
- The output of Low Voltage Detector.

The system fault is valid when FLTyEN bit in SHRTIMER_STxFLTCTL register is set to 1. The system fault can overrides the fault channel input (logic ORed).

19.4.8. Trigger to ADC

The ADCs can be triggered by the Master_TIMER and Slave_TIMERx. Four independent triggers (SHRTIMER_ADCTRIGy,y=0..3) are available to start the routine sequence of the ADCs. Refer to ***Figure 19-46. Trigger to ADC selection overview.***

Figure 19-46. Trigger to ADC selection overview



There are up to 32 events which can be combined (ORed) for each trigger output. They are defined in SHRTIMER_ADCTRIGSy(y=0..3) registers.

SHRTIMER_ADCTRIGSy(y=0..3) registers are preloaded and can be updated synchronously with the timer they are related to. The update source is defined with ADTGyUSRC[2:0] bit-field in the SHRTIMER_CTL0 register. For example, ADTGyUSRC[2:0] = 3'b001 and Slaver_TIMER0 is update source:

- If SHWEN = 1 in SHRTIMER_STxCTL0 register, SHRTIMER_ADCTRIGSy(y=0..3) registers is preloaded and can be updated synchronously with Slaver_TIMER0.
- If SHWEN = 0 in SHRTIMER_STxCTL0 register, SHRTIMER_ADCTRIGSy(y=0..3) registers is not preloaded and a write access will result in an immediate update of the trigger source.

19.4.9. Trigger to DAC

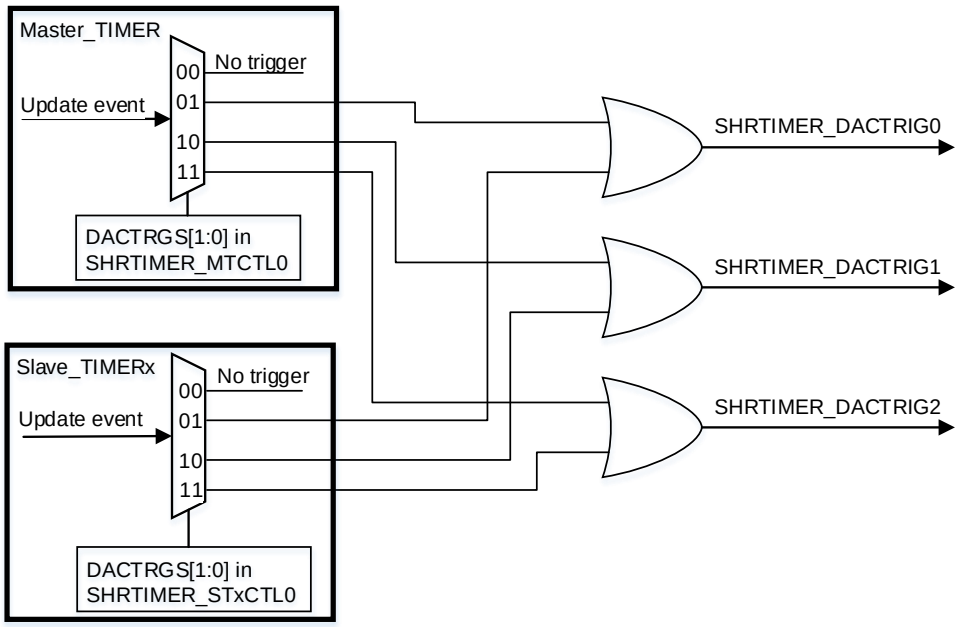
The SHRTIMER allows to have the embedded DACs updated synchronously with the timer updates. The update events from the Master_TIMER and Slave_TIMERx can generate DAC update triggers on SHRTIMER_DACTRIGy(y=0..2).

DACTRGS[1:0] bit-field of the SHRTIMER_MTCTL0 and SHRTIMER_STxCTL0 registers are programmed as follows:

- 00: No DAC trigger event generated
- 01: DAC trigger event generated on SHRTIMER_DACTRIG0
- 10: DAC trigger event generated on SHRTIMER_DACTRIG1
- 11: DAC trigger event generated on SHRTIMER_DACTRIG2

When DACTRGS[1:0] bit-field are enabled in multiple timers, the SHRTIMER_DACTRIGy(y=0..2) will consist of an OR of all timers' update events. Refer to ***Figure 19-47. Trigger to DAC selection overview.***

Figure 19-47. Trigger to DAC selection overview



19.4.10. Interrupt

Most events can generate interrupt requests. All interrupt requests are grouped in 7 vectors (SHRTIMER_IRQy,y=0..6).Refer to ***Table 19-17. Interrupt mapping*** for details

Table 19-17. Interrupt mapping

Interrupt Number	Event	Control bit
Master_TIMER: SHRTIMER_IRQ0	Update event	UPIE in SHRTIMER_MTDMAINTEN
	Synchronization input event	SYNIIE in SHRTIMER_MTDMAINTEN
	Repetition event	REPIE in SHRTIMER_MTDMAINTEN
	Compare 0 event	CMP0IE in SHRTIMER_MTDMAINTEN
	Compare 1 event	CMP1IE in SHRTIMER_MTDMAINTEN
	Compare 2 event	CMP2IE in SHRTIMER_MTDMAINTEN
	Compare 3 event	CMP3IE in SHRTIMER_MTDMAINTEN
Slave_TIMER0: SHRTIMER_IRQ1	Delayed IDLE mode entry	DLYIIE in SHRTIMER_STxDMAINTEN
	Counter reset event	RSTIE in SHRTIMER_STxDMAINTEN
Slave_TIMER1: SHRTIMER_IRQ2	C1OPRE goes from active to inactive	CH1ONAIE in SHRTIMER_STxDMAINTEN
	C1OPRE goes from inactive to active	CH1OAIE in SHRTIMER_STxDMAINTEN
Slave_TIMER2: SHRTIMER_IRQ3	C0OPRE goes from active to inactive	CH0ONAIE in SHRTIMER_STxDMAINTEN
	C0OPRE goes from inactive to active	CH0OAIE in SHRTIMER_STxDMAINTEN
Slave_TIMER3: SHRTIMER_IRQ4	Capture 1 event	CAP1IE in SHRTIMER_STxDMAINTEN
	Capture 0 event	CAP0IE in SHRTIMER_STxDMAINTEN
	Update event	UPIE in SHRTIMER_STxDMAINTEN

Interrupt Number	Event	Control bit
Slave_TIMER4: SHRTIMER_IRQ5	Repetition event	REPIE in SHRTIMER_STxDMAINTEN
	Compare 3 event	CMP3IE in SHRTIMER_STxDMAINTEN
	Compare 2 event	CMP2IE in SHRTIMER_STxDMAINTEN
	Compare 1 event	CMP1IE in SHRTIMER_STxDMAINTEN
	Compare 0 event	CMP0IE in SHRTIMER_STxDMAINTEN
SHRTIMER_IRQ0	Bunch mode period event	BMPERIE in SHRTIMER_INTEN
	DLL calibration completed	DLLCALIE in SHRTIMER_INTEN
SHRTIMER_IRQ6	System fault	SYSFLTIE in SHRTIMER_INTEN
	Fault 4	FLT4IE in SHRTIMER_INTEN
	Fault 3	FLT3IE in SHRTIMER_INTEN
	Fault 2	FLT2IE in SHRTIMER_INTEN
	Fault 1	FLT1IE in SHRTIMER_INTEN
	Fault 0	FLT0IE in SHRTIMER_INTEN

19.4.11. DMA request

Most events can generate DMA requests and each timer corresponds to a DMA channel. Refer to ***Table 19-18. DMA request mapping*** for details.

Table 19-18. DMA request mapping

DMA channel	Event	Control bit
Master_TIMER: DMA0_Channel1	Update event	UPDEN in SHRTIMER_MTDMAINTEN
	Synchronization input event	SYNIDEN in SHRTIMER_MTDMAINTEN
	Repetition event	REPDEN in SHRTIMER_MTDMAINTEN
	Compare 0 event	CMP0DEN in SHRTIMER_MTDMAINTEN
	Compare 1 event	CMP1DEN in SHRTIMER_MTDMAINTEN
	Compare 2 event	CMP2DEN in SHRTIMER_MTDMAINTEN
	Compare 3 event	CMP3DEN in SHRTIMER_MTDMAINTEN
Slave_TIMER0: DMA0_Channel2	Delayed IDLE mode entry	DLYIDEN in SHRTIMER_STxDMAINTEN
	Counter reset event	RSTDEN in SHRTIMER_STxDMAINTEN
Slave_TIMER1: DMA0_Channel3	C1OPRE goes from active to inactive	CH1ONADEN in SHRTIMER_STxDMAINTEN
	C1OPRE goes from inactive to active	CH1OADEN in SHRTIMER_STxDMAINTEN
Slave_TIMER2: DMA0_Channel4	C0OPRE goes from active to inactive	CH0ONADEN in SHRTIMER_STxDMAINTEN
	C0OPRE goes from inactive to active	CH0OADEN in SHRTIMER_STxDMAINTEN
Slave_TIMER3: DMA0_Channel5	Capture 1 event	CAP1DEN in SHRTIMER_STxDMAINTEN
	Capture 0 event	CAP0DEN in SHRTIMER_STxDMAINTEN
	Update event	UPDEN in SHRTIMER_STxDMAINTEN
	Repetition event	REPDEN in SHRTIMER_STxDMAINTEN
Slave_TIMER4: DMA0_Channel6	Compare 3 event	CMP3DEN in SHRTIMER_STxDMAINTEN
	Compare 2 event	CMP2DEN in SHRTIMER_STxDMAINTEN

DMA channel	Event	Control bit
	Compare 1 event	CMP1DEN in SHRTIMER_STxDMAINTEN
	Compare 0 event	CMP0DEN in SHRTIMER_STxDMAINTEN

Note: It is necessary to disable first the DMA controller before disable a DMA request.

19.4.12. DMA mode

Timer's DMA mode is the function that configures SHRTIMER's multiple registers by DMA module with a single DMA request. The relative registers (7 registers in total) are as follows:

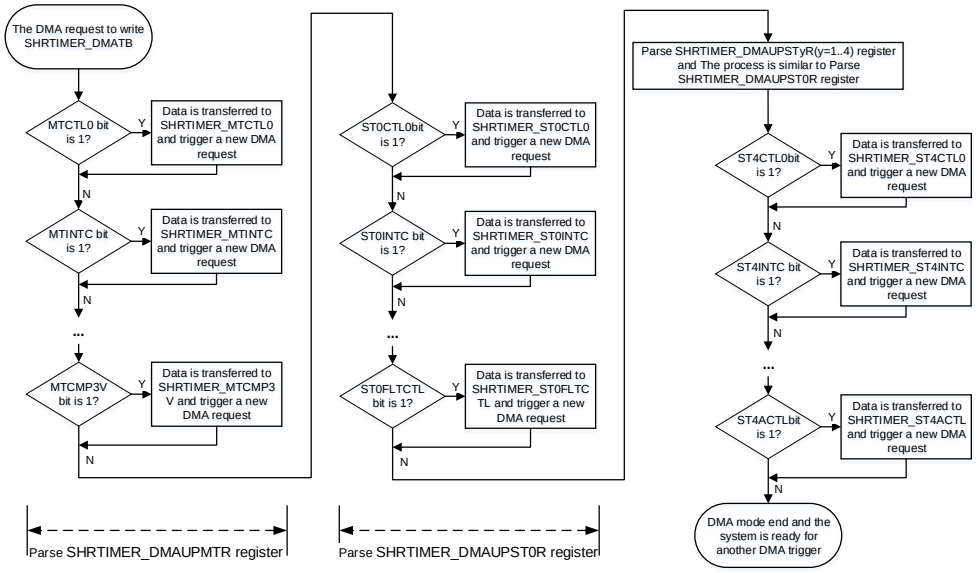
- SHRTIMER_DMAUPMTR: Defines which registers in the Master_TIMER are updated. Most of Master_TIMER control and data registers are associated with a selection bit. If the selection bit is set, the write access is redirected to the associated register.
- SHRTIMER_DMAUPSTxR(x=0..4): Defines which registers in the Slave_TIMERx are updated. Most of Slave_TIMERx control and data registers are associated with a selection bit. If the selection bit is set, the write access is redirected to the associated register.
- SHRTIMER_DMATB: DMA transfer buffer register. It is only necessary to have the DMA module pointing to the SHRTIMER_DMATB register as the destination, to the peripheral configuration with the peripheral increment mode disabled. All write accesses to this register will be internally re-routed to the final destination register by a redirection mechanism.

The DMA mode feature is only available for one DMA channel, but any of the 6 channels can be selected for DMA transfers.

The DMA mode is permanently enabled (there is no enable bit). A DMA operation is started by the first write access into the SHRTIMER_DMATB register.

When the DMA request occurs, the SHRTIMER generates multiple 32-bit DMA requests and parses the register to be updated (defined in SHRTIMER_DMAUPMTR and SHRTIMER_DMAUPSTxR registers). If the selection bit is set to 1, the write access is redirected to the associated register. If the bit is 0, the register update is skipped and the register parsing is resumed until a new bit set is detected, to trigger a new DMA request. Once the six registers (SHRTIMER_DMAUPMTR and SHRTIMER_DMAUPSTxR registers) are all parsed, the DMA mode is completed and a DMA mode end event is generated. The system is ready for another DMA trigger. If one more DMA request event coming, SHRTIMER will repeat the process as above. Refer to ***Figure 19-48. DMA mode operation flowchart.***

Figure 19-48. DMA mode operation flowchart



19.4.13. Debug mode

When the Cortex®-M33 halted, the SHRTIMER_HOLD bit in DBG_CTL0 register determines whether the counter stops running.

SHRTIMER_HOLD = 0

If SHRTIMER_HOLD = 0, the SHRTIMER continues to work normally.

SHRTIMER_HOLD = 1

If SHRTIMER_HOLD = 1, the counter in Master_TIMER and all Slave_TIMERx is stopped.

The outputs enter the Fault state if CHyFLTOS[1:0] = 2’b01, 02’b10, 2’b11. The outputs can be enabled again by settings STxCHyEN bit in SHRTIMER_CHOUTEN register and clearing SHRTIMER_HOLD. If CHyFLTOS[1:0] = 2’b00, the outputs keep their current state. When exit debug mode, the output returns to its original state.

All counter reset/start and capture triggers are disabled. All triggers from external events are disabled, except for triggers to ADCs. The update events are discarded. The bunch mode circuit is frozen: the triggers are ignored and the bunch mode counter stopped.

DLL calibration is run normally. These units which drive the normal output in RUN mode are not affected by debug, for instance the dead-time unit, the carrier-signal, the set/reset crossbar and so on.

19.5. Register definition

SHRTIMER base address: 0x4001 7400

The registers can be segmented for ease of addressing：

SHRTIMER Master_TIMER registers base address: 0x4001 7400

SHRTIMER Slave_TIMER0 registers base address: 0x4001 7480

SHRTIMER Slave_TIMER1 registers base address: 0x4001 7500

SHRTIMER Slave_TIMER2 registers base address: 0x4001 7580

SHRTIMER Slave_TIMER3 registers base address: 0x4001 7600

SHRTIMER Slave_TIMER4 registers base address: 0x4001 7680

SHRTIMER Common registers base address: 0x4001 7780

19.5.1. Master_TIMER registers

SHRTIMER Master_TIMER registers base address: 0x4001 7400

SHRTIMER Master_TIMER control register 0 (SHRTIMER_MTCTL0)

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
UPSEL[1:0]		UPREP	Reserved	SHWEN	DACTRGS[1:0]		Reserved			ST4CEN	ST3CEN	ST2CEN	ST1CEN	ST0CEN	MTCEN
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
SYNOSRC[1:0]		SYNOPLS[1:0]		SYNISTRT	SYNIRST	SYNISRC[1:0]		Reserved		HALFM	CNTRSTM	CTNM	CNTCKDIV[2:0]		
rw		rw		rw	rw	rw				rw	rw	rw	rw		

Bits	Fields	Descriptions
31:30	UPSEL[1:0]	Update event selection This bit-field specifies which the relationship between update events and the DMA mode. 00: Update event generated independently from DMA mode. 01: Update event generated when the DMA transfer is completed in DMA mode. 10: Update event generated on counter roll-over following a DMA transfer completion in DMA mode. This configuration is only applicable in continuous mode. 11: Reserved
29	UPREP	Update event generated by repetition event This bit specifies whether repetition event can generate update event. 0: Update event generated by repetition event disable 1: Update event generated by repetition event enable

		Note: UPREP can be set only if UPSEL[1:0] = 2'b00 or 2'b01.
28	Reserved	Must be kept at reset value
27	SHWEN	Shadow registers enable 0: The shadow registers are disabled 1: The shadow registers are enabled
26:25	DACTRGS[1:0]	Trigger source to DAC The timer can also generate a DAC trigger event when an update event occurs. This bit-field specifies which trigger source generates the DAC trigger event. 00:No DAC trigger event generated 01: DAC trigger event generated on SHRTIMER_DACTRIG0 10: DAC trigger event generated on SHRTIMER_DACTRIG1 11: DAC trigger event generated on SHRTIMER_DACTRIG2
24:22	Reserved	Must be kept at reset value
21	ST4CEN	The counter of Slave_TIMER4 enable 0: The counter of Slave_TIMER4 disable 1: The counter of Slave_TIMER4 enable **Note:** This bit must not be modified within a minimum of 8 $t_{SHRTIMER\_CK}$ clock.
20	ST3CEN	The counter of Slave_TIMER3 enable 0: The counter of Slave_TIMER3 disable 1: The counter of Slave_TIMER3 enable **Note:** This bit must not be modified within a minimum of 8 $t_{SHRTIMER\_CK}$ clock.
19	ST2CEN	The counter of Slave_TIMER2 enable 0: The counter of Slave_TIMER2 disable 1: The counter of Slave_TIMER2 enable **Note:** This bit must not be modified within a minimum of 8 $t_{SHRTIMER\_CK}$ clock.
18	ST1CEN	The counter of Slave_TIMER1 enable 0: The counter of Slave_TIMER1 disable 1: The counter of Slave_TIMER1 enable **Note:** This bit must not be modified within a minimum of 8 $t_{SHRTIMER\_CK}$ clock.
17	ST0CEN	The counter of Slave_TIMER0 enable 0: The counter of Slave_TIMER0 disable 1: The counter of Slave_TIMER0 enable **Note:** This bit must not be modified within a minimum of 8 $t_{SHRTIMER\_CK}$ clock.
16	MTCEN	The counter of Master_TIMER enable 0: The counter of Master_TIMER disable 1: The counter of Master_TIMER enable **Note:** This bit must not be modified within a minimum of 8 $t_{SHRTIMER\_CK}$ clock.
15:14	SYNOSRC[1:0]	Synchronization output source

		This bit-field specifies the event to be sent to the synchronization output pad SHRTIMER_SCOUT. 00: Master_TIMER start event. 01: Master_TIMER compare 0 event 10: Slave_TIMER0 reset and start event 11: Slave_TIMER0 compare 0 event
13:12	SYNOPLS[1:0]	Synchronization output pulse This bit-field specifies the pulse on the synchronization output pad SHRTIMER_SCOUT. 00: Pulse generated disable. No pulse on SHRTIMER_SCOUT. 01: Reserved. 10: Positive pulse generated on the SHRTIMER_SCOUT. The length of it is 16 $t_{SHRTIMER\_CK}$ cycles. 11: Negative pulse generated on the SHRTIMER_SCOUT. The length of it is 16 $t_{SHRTIMER\_CK}$ cycles.
11	SYNISTRT	Synchronization input start counter This bit specifies whether the synchronous input can start the counter. 0: Synchronization input cannot start counter. 1: Synchronization input can start counter.
10	SYNIRST	Synchronization input reset counter This bit specifies whether the synchronous input can reset the counter. 0: Synchronization input cannot reset counter. 1: Synchronization input can reset counter.
9:8	SYNISRC[1:0]	Synchronization input source This bit-field specifies the synchronization input source. 00: Synchronization input disable. 01: Reserved. 10: Internal signal: TIMER0_TRGO in the advanced timer TIMER0 11: External signal: a positive pulse on the SHRTIMER_SCIN pin triggers the Master_TIMER. **Note:** This bit-field cannot be modified once the impacted timers are enabled
7:6	Reserved	Must be kept at reset value
5	HALFM	Half mode When the bit is set, SHRTIMER_MTCMP0V active register is always the half of counter auto-reload value (SHRTIMER_MTCAR). 0: Half mode disable. 1: Half mode enable.
4	CNTRSTM	Counter reset mode This bit defines the behavior of the timer counter in single pulse mode. 0: The counter can be reset only if it stops (period elapsed)

		1: The counter can be reset at any time (running or stopped).
3	CTNM	Continuous mode 0: Single pulse mode. The counter stops by hardware when it reaches the SHRTIMER_MTCAR value. 1: Continuous mode. The counter rolls over to zero and count continuously when it reaches the SHRTIMER_MTCAR value
2:0	CNTCKDIV[2:0]	Counter clock division This bit-field can be configured by software to specify division ratio between the super high resolution clock (SHRTIMER_HPCK) and the counter clock (SHRTIMER_PSCCK). When the CNTCKDIV[3] is '0', $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/2^{CNTCKDIV[2:0]+1}$. When the CNTCKDIV[3] bit in the SHRTIMER_MTACTL is '1' and CNTCKDIV[2:0] can only be configured with '3'b000': $f_{SHRTIMER\_PSSCK} = f_{SHRTIMER\_HPCK}$ 0000: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/2$ 0001: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/4$ 0010: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/8$ 0011: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/16$ 0100: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/32$ 0101: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/64$ 0110: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/128$ 0111: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/256$ 1000: $f_{SHRTIMER\_PSSCK} = f_{SHRTIMER\_HPCK}$ Other values are reserved. **Note:** The CNTCKDIV[3:0] bit-field cannot be modified once the timer is enabled.

SHRTIMER Master_TIMER interrupt flag register (SHRTIMER_MTINTF)

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									UPIF	SYNIIF	REPIF	CMP3IF	CMP2IF	CMP1IF	CMP0IF
									r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value
6	UPIF	Update interrupt flag

This flag is set by hardware when an update event occurs.
0: No update interrupt occurred
1: Update interrupt occurred

5	SYNIIF	Synchronization input interrupt flag This flag is set by hardware when synchronization input occurs. 0: No synchronization input interrupt occurred 1: Synchronization input interrupt occurred
4	REPIF	Repetition interrupt flag This flag is set by hardware when a repetition event occurs. 0: No repetition interrupt occurred 1: Repetition interrupt occurred
3	CMP3IF	Compare 3 interrupt flag This flag is set by hardware when a compare 3 event occurs. 0: No compare 3 interrupt occurred 1: Compare 3 interrupt occurred
2	CMP2IF	Compare 2 interrupt flag This flag is set by hardware when a compare 2 event occurs. 0: No compare 2 interrupt occurred 1: Compare 2 interrupt occurred
1	CMP1IF	Compare 1 interrupt flag This flag is set by hardware when a compare 1 event occurs. 0: No compare 1 interrupt occurred 1: Compare 1 interrupt occurred
0	CMP0IF	Compare 0 interrupt flag This flag is set by hardware when a compare 0 event occurs. 0: No compare 0 interrupt occurred 1: Compare 0 interrupt occurred

SHRTIMER Master_TIMER interrupt flag clear register (SHRTIMER_MTINTC)

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									UPIFC	SYNIIFC	REPIFC	CMP3IFC	CMP2IFC	CMP1IFC	CMP0IFC

			w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value
6	UPIFC	Clear update interrupt flag 0: No effect 1: Clear update interrupt flag
5	SYNIIFC	Clear synchronization input interrupt flag 0: No effect 1: Clear synchronization input interrupt flag
4	REPIFC	Clear repetition interrupt flag 0: No effect 1: Clear repetition interrupt flag
3	CMP3IFC	Clear compare 3 interrupt flag 0: No effect 1: Clear compare 3 interrupt flag
2	CMP2IFC	Clear compare 2 interrupt flag 0: No effect 1: Clear compare 2 interrupt flag
1	CMP1IFC	Clear compare 1 interrupt flag 0: No effect 1: Clear compare 1 interrupt flag
0	CMP0IFC	Clear compare 0 interrupt flag 0: No effect 1: Clear compare 0 interrupt flag

SHRTIMER Master_TIMER DMA and interrupt enable register (SHRTIMER_MTDMAINTEN)

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									UPDEN	SYNIDEN	REPDEN	CMP3DEN	CMP2DEN	CMP1DEN	CMP0DEN
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									UPIE	SYNIIE	REPIE	CMP3IE	CMP2IE	CMP1IE	CMP0IE

rw rw rw rw rw rw rw

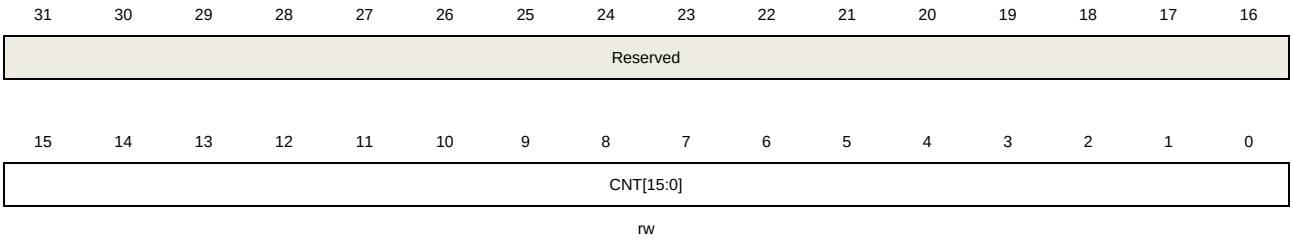
Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value
22	UPDEN	Update DMA request enable 0: disabled 1: enabled
21	SYNIDEN	Synchronization input DMA request enable 0: disabled 1: enabled
20	REPDEN	Repetition DMA request enable 0: disabled 1: enabled
19	CMP3DEN	Compare 3 DMA request enable 0: disabled 1: enabled
18	CMP2DEN	Compare 2 DMA request enable 0: disabled 1: enabled
17	CMP1DEN	Compare 1 DMA request enable 0: disabled 1: enabled
16	CMP0DEN	Compare 0 DMA request enable 0: disabled 1: enabled
15:7	Reserved	Must be kept at reset value
6	UPIE	Update interrupt enable 0: disabled 1: enabled
5	SYNIIE	Synchronization input interrupt enable 0: disabled 1: enabled
4	REPIE	Repetition interrupt enable 0: disabled 1: enabled
3	CMP3IE	Compare 3 interrupt enable 0: disabled

		1: enabled
2	CMP2IE	Compare 2 interrupt enable 0: disabled 1: enabled
1	CMP1IE	Compare 1 interrupt enable 0: disabled 1: enabled
0	CMP0IE	Compare 0 interrupt enable 0: disabled 1: enabled

SHRTIMER Master_TIMER counter register (SHRTIMER_MTCNT)

Address offset: 0x10
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CNT[15:0]	The current counter value. Writing to it can change the value of the counter only when the Master_TIMER is stopped (MTCEN = 0 in SHRTIMER_MTCTL0 register). **Note:** (1) For counter clock division below 64 (CNTCKDIV[3:0] < 5), the least significant bits of the counter are not significant. They cannot be written and read 0. (2) If the value written to this bit-field is above the SHRTIMER_MPER register value, the behavior of the timer is unpredictable.

SHRTIMER Master_TIMER counter auto reload register (SHRTIMER_MTCAR)

Address offset: 0x14
Reset value: 0x0000 FFDF

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-field specifies the auto reload value of the counter. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register. **Note:** (1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000. (2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Master_TIMER counter repetition register (SHRTIMER_MTCREP)

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								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-field specifies the repetition event generation rate. When the repetition counter had count down to zero, the coming roll-over event in continuous mode or reset event will generate a repetition event. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the

shadow register.

SHRTIMER Master_TIMER compare 0 value register (SHRTIMER_MTCMP0V)

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
CMP0VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMP0VAL[15:0]	Compare 0 value This bit-field contains value to be compared to the counter. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register. **Note:** (1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000. (2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Master_TIMER compare 1 value register (SHRTIMER_MTCMP1V)

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
CMP1VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMP1VAL[15:0]	Compare 1 value This bit-field contains value to be compared to the counter. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register. **Note:** (1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000. (2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Master_TIMER compare 2 value register (SHRTIMER_MTCMP2V)

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
CMP2VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMP2VAL[15:0]	Compare 2 value This bit-field contains value to be compared to the counter. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register. **Note:** (1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000. (2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Master_TIMER compare 3 value register (SHRTIMER_MTCMP3V)

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
CMP3VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMP3VAL[15:0]	Compare 3 value This bit-field contains value to be compared to the counter. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register. **Note:** (1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000. (2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Master_TIMER additional control register (SHRTIMER_MTACTL)

Address offset: 0x7C
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												CNTCKDIV[3]	Reserved		
												rw			

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3	CNTCKDIV[3]	Counter clock division This bit-field can be configured by software to specify division ratio between the

		super high resolution clock (SHRTIMER_HPCK) and the counter clock (SHRTIMER_PSCCK). When the CNTCKDIV[3] is '0', $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/2^{CNTCKDIV[2:0]+1}$. When the CNTCKDIV[3] bit in the SHRTIMER_MTACTL is '1' and CNTCKDIV[2:0] can only be configured with '3'b000': $f_{SHRTIMER\_PSSCK} = f_{SHRTIMER\_HPCK}$ **Note:** The CNTCKDIV[3:0] bit-field cannot be modified once the timer is enabled
2:0	Reserved	Must be kept at reset value

19.5.2. Slave_TIMERx registers(x=0..4)

SHRTIMER Slave_TIMER0 registers base address: 0x4001 7480

SHRTIMER Slave_TIMER1 registers base address: 0x4001 7500

SHRTIMER Slave_TIMER2 registers base address: 0x4001 7580

SHRTIMER Slave_TIMER3 registers base address: 0x4001 7600

SHRTIMER Slave_TIMER4 registers base address: 0x4001 7680

SHRTIMER Slave_TIMERx control register 0 (SHRTIMER_STxCTL0)

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
UPSEL[3:0]				SHWEN	DACTRGS[1:0]		UPBMT	UPBST4	UPBST3	UPBST2	UPBST1	UPBST0	UPRST	UPREP	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
DELCMP3M[1:0]		DELCMP1M[1:0]		SYNISTRT	SYNIRST	Reserved			BLNMEN	HALFM	CNTRSTM	CTNM	CNTCKDIV[2:0]		
rw		rw		rw	rw				rw	rw	rw	rw	rw		

Bits	Fields	Descriptions
31:28	UPSEL[3:0]	Update event selection This bit-field specifies which the relationship between update events and the DMA mode. 0000: Update event generated independently from DMA mode. 0001: Update event generated when the DMA transfer completed in DMA mode. 0010: Update event generated on the update event following the DMA transfer completed in DMA mode. 0011: Update event generated on the rising edge of STxUPIN0. 0100: Update event generated on the rising edge of STxUPIN1. 0101: Update event generated on the rising edge of STxUPIN2.

		0110: Update event generated on the update event following the rising edge of STxUPIN0. 0111: Update event generated on the update event following the rising edge of STxUPIN1. 1000: Update event generated on the update event following the rising edge of STxUPIN2. Other values are reserved **Note:** (1) The bit-field must reset before writing new value. (2) When UPSEL[3:0] = 4'b0001, 4'b0011, 4'b0100, 4'b0101, it is possible to have multiple concurrent update source. For instance, update by Master_TIMER (UPBMT = 1) and DMA mode.
27	SHWEN	Shadow registers enable 0: The shadow registers are disabled 1: The shadow registers are enabled
26:25	DACTRGS[1:0]	Trigger source to DAC The timer can also generate a DAC trigger event when an update event occurs. This bit-field specifies which trigger source generates the DAC trigger event. 00:No DAC trigger event generated 01: DAC trigger event generated on SHRTIMER_DACTRIG0 10: DAC trigger event generated on SHRTIMER_DACTRIG1 11: DAC trigger event generated on SHRTIMER_DACTRIG2
24	UPBMT	Update by Master_TIMER update event When the bit is set, the Slave_TIMERx(x=0..4) update event are synchronized with Master_TIMER update event and the active registers of them are updated by the Master_TIMER update event 0: The active registers is not update by Master_TIMER. 1: The active registers is update by Master_TIMER.
23	UPBST4	Update by Slave_TIMER4 update event When the bit is set, the Slave_TIMERx(x=0..3) update event are synchronized with Slave_TIMER4 update event and the active registers of them are updated by the Slave_TIMER4 update event 0: The active registers is not update by Slave_TIMER4. 1: The active registers is update by Slave_TIMER4. **Note:** This bit does not exist in Slave_TIMER4.
22	UPBST3	Update by Slave_TIMER3 update event When the bit is set, the Slave_TIMERx(x=0..2,4) update event are synchronized with Slave_TIMER3 update event and the active registers of them are updated by the Slave_TIMER3 update event 0: The active registers is not update by Slave_TIMER3. 1: The active registers is update by Slave_TIMER3.

		Note: This bit does not exist in Slave_TIMER3.
21	UPBST2	Update by Slave_TIMER2 update event When the bit is set, the Slave_TIMERx(x=0,1,3,4) update event are synchronized with Slave_TIMER2 update event and the active registers of them are updated by the Slave_TIMER2 update event 0: The active registers is not update by Slave_TIMER2. 1: The active registers is update by Slave_TIMER2. **Note:** This bit does not exist in Slave_TIMER2.
20	UPBST1	Update by Slave_TIMER1 update event When the bit is set, the Slave_TIMERx(x=0,2,3,4) update event are synchronized with Slave_TIMER1 update event and the active registers of them are updated by the Slave_TIMER1 update event 0: The active registers is not update by Slave_TIMER1. 1: The active registers is update by Slave_TIMER1. **Note:** This bit does not exist in Slave_TIMER1.
19	UPBST0	Update by Slave_TIMER0 update event When the bit is set, the Slave_TIMERx(x=1..4) update event are synchronized with Slave_TIMER0 update event and the active registers of them are updated by the Slave_TIMER0 update event 0: The active registers is not update by Slave_TIMER0. 1: The active registers is update by Slave_TIMER0. **Note:** This bit does not exist in Slave_TIMER0.
18	UPRST	Update event generated by reset event This bit specifies whether counter reset event or roll-over event can generate update event. 0: Update event generated by reset event or roll-over event disable 1: Update event generated by reset event or roll-over event enable
17	UPREP	Update event generated by repetition event This bit specifies whether repetition event can generate update event. 0: Update event generated by repetition event disable 1: Update event generated by repetition event enable
16	Reserved	Must be kept at reset value
15:14	DELCMP3M[1:0]	Compare 3 delayed mode 00: Compare 3 delayed mode disable. Compare match occurs as soon as the counter equals the value of compare 3 active register. 01: Compare 3 delayed mode 0. After a capture 1 event, the recalculated value of compare 3 is: (compare 3 active register value + capture 1 value). Compare match occurs as soon as the counter equals the recalculated value. 10: Compare 3 delayed mode 1. After a capture 1 event or compare 0 event, the recalculated value of compare 3 is: (compare 3 active register value + capture 1

		value for capture 1 event, or compare 3 active register value + compare 0 value for compare 0 event). Compare match occurs as soon as the counter equals the recalculated value. 11: Compare 3 delayed mode 2. After a capture 1 event or compare 2 event, the recalculated value of compare 3 is: (compare 3 active register value + capture 1 value for capture 1 event, or compare 3 active register value + compare 2 value for compare 2 event). Compare match occurs as soon as the counter equals the recalculated value. **Note:** This bit-field must not be modified once the counter is enabled (STxCEN =1 in SHRTIMER_MTCTL0 register).
13:12	DELCMP1M[1:0]	Compare 1 delayed mode 00: Compare 1 delayed mode disable. Compare match occurs as soon as the counter equals the value of compare 1 active register. 01: Compare 1 delayed mode 0. After a capture 0 event, the recalculated value of compare 1 is: (compare 1 active register value + capture 0 value). Compare match occurs as soon as the counter equals the recalculated value. 10: Compare 1 delayed mode 1. After a capture 0 event or compare 0 event, the recalculated value of compare 1 is: (compare 1 active register value + capture 0 value for capture 0 event, or compare 1 active register value + compare 0 active register value for compare 0 event). Compare match occurs as soon as the counter equals the recalculated value. 11: Compare 1 delayed mode 2. After a capture 0 event or compare 2 event, the recalculated value of compare 1 is:(compare 1 active register value + capture 0 value for capture 0 event, or compare 1 active register value + compare 2 value for compare 2 event). Compare match occurs as soon as the counter equals the recalculated value. **Note:** This bit-field must not be modified once the counter is enabled (STxCEN =1 in SHRTIMER_MTCTL0 register).
11	SYNISTRT	Synchronous input start timer This bit specifies whether the synchronous input signal can start the counter 0: The synchronous input signal cannot start the counter 1: The synchronous input signal can start the counter
10	SYNIRST	Synchronous input reset timer This bit specifies whether the synchronous input signal can reset the counter 0: The synchronous input signal cannot reset the counter 1: The synchronous input signal can reset the counter
9:7	Reserved	Must be kept at reset value
6	BLNMEN	Balanced mode enable 0: Balanced mode disable 1: Balanced mode enable **Note:** This bit-field must not be modified once the counter is enabled (STxCEN =1

		in SHRTIMER_MTCTL0 register).
5	HALFM	Half mode When the bit is set, SHRTIMER_STxCMP0V active register is always the half of counter auto-reload value (SHRTIMER_STxCAR). 0: Half mode disable. 1: Half mode enable.
4	CNTRSTM	Counter reset mode This bit defines the behavior of the timer counter in single pulse mode. 0: The counter can be reset only if it stops (period elapsed) 1: The counter can be reset at any time (running or stopped).
3	CTNM	Continuous mode. 0: Single pulse mode. The counter stops by hardware when it reaches the SHRTIMER_STxCAR value. 1: Continuous mode. The counter rolls over to zero and count continuously when it reaches the SHRTIMER_STxCAR value
2:0	CNTCKDIV[2:0]	Counter clock division This bit-field can be configured by software to specify division ratio between the super high resolution clock (SHRTIMER_HPCK) and the counter clock (SHRTIMER_PSCCK). When the CNTCKDIV[3] is '0', $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/2^{CNTCKDIV[2:0]+1}$. When the CNTCKDIV[3] bit in the SHRTIMER_STxACTL is '1' and CNTCKDIV[2:0] can only be configured with '3'b000': $f_{SHRTIMER\_PSSCK} = f_{SHRTIMER\_HPCK}$ 0000: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/2$ 0001: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/4$ 0010: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/8$ 0011: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/16$ 0100: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/32$ 0101: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/64$ 0110: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/128$ 0111: $f_{SHRTIMER\_PSCCK} = f_{SHRTIMER\_HPCK}/256$ 1000: $f_{SHRTIMER\_PSSCK} = f_{SHRTIMER\_HPCK}$ Other values are reserved. **Note:** This bit-field must not be modified once the counter is enabled (STxCEN =1 in SHRTIMER_MTCTL0 register).

SHRTIMER Slave_TIMERx interrupt flag register (SHRTIMER_STxINTF)

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										CH1F	CH0F	Reserved		BLNIF	CBLNF
										r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DLYIIF	RSTIF	CH1ONAIF	CH1OAIF	CH0ONAIF	CH0OAIF	CAP1IF	CAP0IF	UPIF	Reserved	REPIF	CMP3IF	CMP2IF	CMP1IF	CMP0IF
	r	r	r	r	r	r	r	r	r		r	r	r	r	r

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21	CH1F	Channel 1 output flag This bit indicates the output level state of channel 1. 0: Channel 1 outputs inactive level. 1: Channel 1 outputs active level.
20	CH0F	Channel 0 output flag This bit indicates the output level state of channel 0. 0: Channel 0 outputs inactive level. 1: Channel 0 outputs active level.
19:18	Reserved	Must be kept at reset value
17	BLNIF	Balanced IDLE flag This bit indicates which channel is outputting the signal when balanced IDLE entry occurred. 0: Channel 0 outputs the signal and channel 1 outputs inactive level when balanced IDLE entry occurred. 1: Channel 1 outputs the signal and channel 0 outputs inactive level when balanced IDLE entry occurred.
16	CBLNF	Current balanced flag This bit is only valid in balanced mode. This bit indicates which channel is currently outputting the signal. 0: Channel 0 outputs the signal and channel 1 outputs inactive level. 1: Channel 1 outputs the signal and channel 0 outputs inactive level.
15	Reserved	Must be kept at reset value
14	DLYIIF	Delayed IDLE mode entry interrupt flag This flag is set by hardware when delayed IDLE or balanced IDLE mode entry occurred. 0: No counter delayed IDLE mode entry interrupt occurred 1: Delayed IDLE mode entry interrupt occurred
13	RSTIF	Counter reset interrupt flag This flag is set by hardware when counter reset or roll-over event occurred.

		0: No counter reset or roll-over interrupt occurred 1: Counter reset or roll-over interrupt occurred
12	CH1ONAIF	Channel 1 output inactive interrupt flag Refer to CH0ONAIF description.
11	CH1OAIF	Channel 1 output active interrupt flag Refer to CH0OAIF description.
10	CH0ONAIF	Channel 0 output inactive interrupt flag This flag is set by hardware when channel 0 output inactive (C0OPRE from active to inactive) occurs. 0: No channel 0 output inactive interrupt occurred 1: Channel 0 output inactive interrupt occurred
9	CH0OAIF	Channel 0 output active interrupt flag This flag is set by hardware when channel 0 output active (C1OPRE from inactive to active) occurs. 0: No channel 0 output active interrupt occurred 1: Channel 0 output active interrupt occurred
8	CAP1IF	Capture 1 interrupt flag This flag is set by hardware when capture 1 event occurs. 0: No capture 1 interrupt occurred 1: Capture 1 interrupt occurred
7	CAP0IF	Capture 0 interrupt flag This flag is set by hardware when capture 0 event occurs. 0: No capture 0 interrupt occurred 1: Capture 0 interrupt occurred
6	UPIF	Update interrupt flag This flag is set by hardware when an update event occurs. 0: No update interrupt occurred 1: Update interrupt occurred
5	Reserved	Must be kept at reset value
4	REPIF	Repetition interrupt flag This flag is set by hardware when a repetition event occurs. 0: No repetition interrupt occurred 1: Repetition interrupt occurred
3	CMP3IF	Compare 3 interrupt flag This flag is set by hardware when a compare 3 event occurs. 0: No compare 3 interrupt occurred 1: Compare 3 interrupt occurred
2	CMP2IF	Compare 2 interrupt flag

		This flag is set by hardware when a compare 2 event occurs. 0: No compare 2 interrupt occurred 1: Compare 2 interrupt occurred
1	CMP1IF	Compare 1 interrupt flag This flag is set by hardware when a compare 1 event occurs. 0: No compare 1 interrupt occurred 1: Compare 1 interrupt occurred
0	CMP0IF	Compare 0 interrupt flag This flag is set by hardware when a compare 0 event occurs. 0: No compare 0 interrupt occurred 1: Compare 0 interrupt occurred

SHRTIMER Slave_TIMERx interrupt flag clear register (SHRTIMER_STxINTC)

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	DLYIIFC	RSTIFC	CH1ONAIFC	CH1OAIFC	CH0ONAIFC	CH0OAIFC	CAP1IFC	CAP0IFC	UPIFC	Reserved	REPIFC	CMP3IFC	CMP2IFC	CMP1IFC	CMP0IFC
	w	w	w	w	w	w	w	w	w		w	w	w	w	w

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	DLYIIFC	Clear delayed IDLE mode entry interrupt flag 0: No effect 1: Clear delayed IDLE state mode entry interrupt flag (DLYIIF in SHRTIMER_STxINTF register)
13	RSTIFC	Clear counter reset interrupt flag 0: No effect 1: Clear counter reset interrupt flag (RSTIF in SHRTIMER_STxINTF register)
12	CH1ONAIFC	Clear channel 1 output inactive interrupt flag Clear CH1ONAIF in SHRTIMER_STxINTF register. Refer to CH0ONAIFC description.
11	CH1OAIFC	Clear channel 1 output active interrupt flag

		Clear CH1OAIF in SHRTIMER_STxINTF register. Refer to CH0OAIFC description.
10	CH0ONAIFC	Clear channel 0 output inactive interrupt flag 0: No effect 1: Clear channel 0 output inactive interrupt flag (CH0ONAIF in SHRTIMER_STxINTF register)
9	CH0OAIFC	Clear channel 0 output active interrupt flag 0: No effect 1: Clear channel 0 output inactive interrupt flag (CH0OAIF in SHRTIMER_STxINTF register)
8	CAP1IFC	Clear capture 1 interrupt flag 0: No effect 1: Clear capture 1 interrupt flag (CAP1IF in SHRTIMER_STxINTF register)
7	CAP0IFC	Clear capture 0 interrupt flag 0: No effect 1: Clear capture 0 interrupt flag (CAP0IF in SHRTIMER_STxINTF register)
6	UPIFC	Clear update interrupt flag 0: No effect 1: Clear update interrupt flag (UPIF in SHRTIMER_STxINTF register)
5	Reserved	Must be kept at reset value
4	REPIFC	Clear repetition interrupt flag 0: No effect 1: Clear repetition interrupt flag (REPIF in SHRTIMER_STxINTF register)
3	CMP3IFC	Clear compare 3 interrupt flag 0: No effect 1: Clear compare 3 interrupt flag (CMP3IF in SHRTIMER_STxINTF register)
2	CMP2IFC	Clear compare 2 interrupt flag 0: No effect 1: Clear compare 2 interrupt flag (CMP2IF in SHRTIMER_STxINTF register)
1	CMP1IFC	Clear compare 1 interrupt flag 0: No effect 1: Clear compare 1 interrupt flag (CMP1IF in SHRTIMER_STxINTF register)
0	CMP0IFC	Clear compare 0 interrupt flag 0: No effect 1: Clear compare 0 interrupt flag (CMP0IF in SHRTIMER_STxINTF register)

SHRTIMER Slave_TIMERx DMA and interrupt enable register (SHRTIMER_STxDMAINTEN)

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	DLYIDEN	RSTDEN	CH1ONADEN	CH1OADEN	CH0ONADEN	CH0OADEN	CAP1DEN	CAP0DEN	UPDEN	Reserved	REPDEN	CMP3DEN	CMP2DEN	CMP1DEN	CMP0DEN
	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
Reserved	DLYIIE	RSTIE	CH1ONAIE	CH1OAIE	CH0ONAIE	CH0OAIE	CAP1IE	CAP0IE	UPIE	Reserved	REPIE	CMP3IE	CMP2IE	CMP1IE	CMP0IE
	rw	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	DLYIDEN	Delayed IDLE mode entry DMA request enable 0: disabled 1: enabled
29	RSTDEN	Counter reset DMA request enable 0: disabled 1: enabled
28	CH1ONADEN	Channel 1 output inactive DMA request enable Refer to CH0ONADEN description.
27	CH1OADEN	Channel 1 output active DMA request enable Refer to CH0OADEN description.
26	CH0ONADEN	Channel 0 output inactive DMA request enable 0: disabled 1: enabled
25	CH0ADEN	Channel 0 output active DMA request enable 0: disabled 1: enabled
24	CAP1DEN	Capture 1 DMA request enable 0: disabled 1: enabled
23	CAP0DEN	Capture 0 DMA request enable 0: disabled

		1: enabled
22	UPDEN	Update DMA request enable 0: disabled 1: enabled
21	Reserved	Must be kept at reset value
20	REPDEN	Repetition DMA request enable 0: disabled 1: enabled
19	CMP3DEN	Compare 3 DMA request enable 0: disabled 1: enabled
18	CMP2DEN	Compare 2 DMA request enable 0: disabled 1: enabled
17	CMP1DEN	Compare 1 DMA request enable 0: disabled 1: enabled
16	CMP0DEN	Compare 0 DMA request enable 0: disabled 1: enabled
15	Reserved	Must be kept at reset value
14	DLYIIE	Delayed IDLE mode entry interrupt enable 0: disabled 1: enabled
13	RSTIE	Counter reset interrupt enable 0: disabled 1: enabled
12	CH1ONAIE	Channel 1 output inactive interrupt enable Refer to CH0ONAIE description.
11	CH1OAIE	Channel 1 output active interrupt enable Refer to CH0OAIE description.
10	CH0ONAIE	Channel 0 output inactive interrupt enable 0: disabled 1: enabled
9	CH0OAIE	Channel 0 output active interrupt enable 0: disabled

		1: enabled
8	CAP1IE	Capture 1 interrupt enable 0: disabled 1: enabled
7	CAP0IE	Capture 0 interrupt enable 0: disabled 1: enabled
6	UPIE	Update interrupt enable 0: disabled 1: enabled
5	Reserved	Must be kept at reset value
4	REPIE	Repetition interrupt enable 0: disabled 1: enabled
3	CMP3IE	Compare 3 interrupt enable 0: disabled 1: enabled
2	CMP2IE	Compare 2 interrupt enable 0: disabled 1: enabled
1	CMP1IE	Compare 1 interrupt enable 0: disabled 1: enabled
0	CMP0IE	Compare 0 interrupt enable 0: disabled 1: enabled

SHRTIMER Slave_TIMERx counter register (SHRTIMER_STxCNT)

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
CNT[15:0]															

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CNT[15:0]	The current counter value. Writing to it can change the value of the counter only when the Slave_TIMERx is stopped (STxCEN = 0 in SHRTIMER_STxCTL0 register). **Note:** (1) For counter clock division below 64 (CNTCKDIV[3:0] < 5), the least significant bits of the counter are not significant. They cannot be written and read 0. (2) If the value written to this bit-field is above the SHRTIMER_MPER register value, the behavior of the timer is unpredictable.

SHRTIMER Slave_TIMERx counter auto reload register (SHRTIMER_STxCAR)

Address offset: 0x14
Reset value: 0x0000 FFDF

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]															

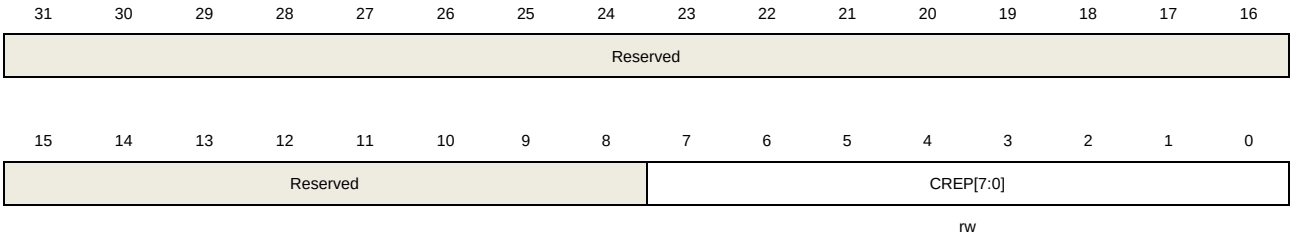
rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CARL[15:0]	Counter auto reload value This bit-field specifies the auto reload value of the counter. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register. **Note:** (1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000. (2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Slave_TIMERx counter repetition register (SHRTIMER_STxCREP)

Address offset: 0x18
Reset value: 0x0000 0000

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

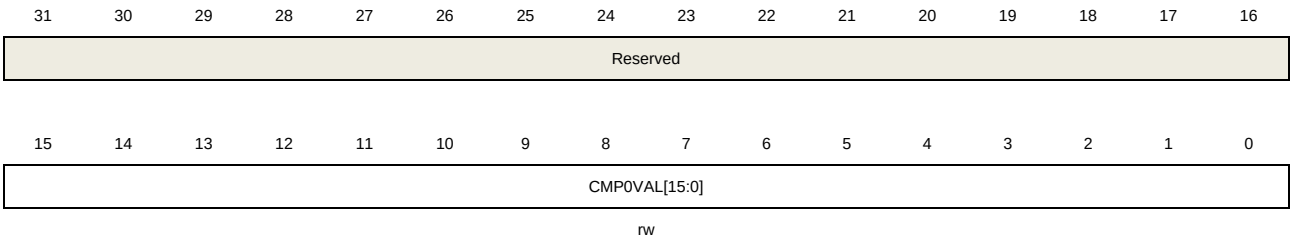


Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	CREP[7:0]	Counter repetition value This bit-field specifies the repetition event generation rate. When the repetition counter had count down to zero, the coming roll-over event in continuous mode or reset event will generate a repetition event. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register.

SHRTIMER Slave_TIMERx compare 0 value register (SHRTIMER_STxCMP0V)

Address offset: 0x1C
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMP0VAL[15:0]	Compare 0 value This bit-field contains value to be compared to the counter. This register has a shadow register. If the shadow register is disabled (SHWEN =

0), it holds the content of the active register; otherwise, it holds the content of the shadow register.

Note:

(1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000.

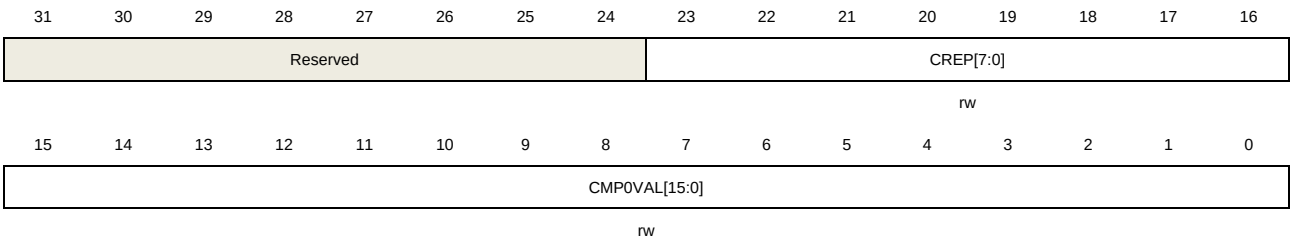
(2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Slave_TIMERx compare 0 composite register (SHRTIMER_STxCMP0CP)

Address offset: 0x20

Reset value: 0x0000 0000

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



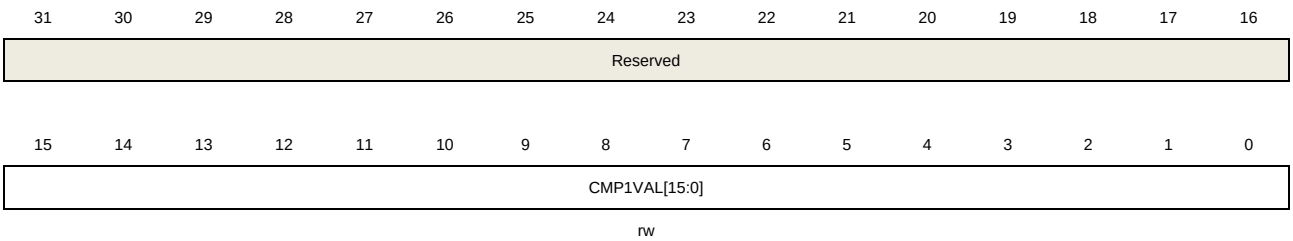
Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:16	CREP[7:0]	Counter repetition value This bit-field is an alias from the CREP[7:0] in the SHRTIMER_STxCREP
15:0	CMP0VAL[15:0]	Compare 0 value This bit-field is an alias from the CMP0VAL[15:0] in the SHRTIMER_STxCMP0V register

SHRTIMER Slave_TIMERx compare 1 value register (SHRTIMER_STxCMP1V)

Address offset: 0x24

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMP1VAL[15:0]	Compare 1 value This bit-field contains value to be compared to the counter. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register. In delayed mode, the active register can be recalculated. **Note:** (1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000. (2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Slave_TIMERx compare 2 value register (SHRTIMER_STxCMP2V)

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
CMP2VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMP2VAL[15:0]	Compare 2 value This bit-field contains value to be compared to the counter. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register. **Note:** (1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000. (2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Slave_TIMERx compare 3 value register (SHRTIMER_STxCMP3V)

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
CMP3VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMP3VAL[15:0]	Compare 3 value This bit-field contains value to be compared to the counter. This register has a shadow register. If the shadow register is disabled (SHWEN = 0), it holds the content of the active register; otherwise, it holds the content of the shadow register. In delayed mode, the active register can be recalculated. **Note:** (1) The minimum value must be greater than or equal to 3 $t_{SHRTIMER\_CK}$. For example: CARL[15:0] >= 0x60 when CNTCKDIV[3:0] = 4'b0000. (2) The maximum value must be less than or equal to 0xFFFF – (1 $t_{SHRTIMER\_CK}$). For example: CARL[15:0] <= 0xFFDF when CNTCKDIV[3:0] = 4'b0000.

SHRTIMER Slave_TIMERx capture 0 value register (SHRTIMER_STxCAP0V)

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
CAP0VAL[15:0]															
r															

Bits	Fields	Descriptions

31:16	Reserved	Must be kept at reset value
15:0	CAP0VAL[15:0]	Capture 0 value This bit-field indicates the counter value corresponding to the last capture event. And this bit-field is read-only. **Note:** For counter clock division below 64 (CNTCKDIV[3:0] < 5), the least significant bits of the counter are not significant. They cannot be written and read 0.

SHRTIMER Slave_TIMERx capture 1 value register (SHRTIMER_STxCAP1V)

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
CAP1VAL[15:0]															
r															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CAP1VAL[15:0]	Capture 1 value This bit-field indicates the counter value corresponding to the last capture event. And this bit-field is read-only. **Note:** For counter clock division below 32 (CNTCKDIV[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and read 0.

SHRTIMER Slave_TIMERx dead-time control register (SHRTIMER_STxDTCTL)

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
DTFSVPROT	DTFSPROT	Reserved				DTFS	DTFCFG[8:0]								
rwo	rwo					rw	rw								

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

DTRSVPROT	DTRSPROT	DTGCKDIV[3:0]	DTRS	DTRCFG[8:0]
rwo	rwo	rw	rw	rw

Bits	Fields	Descriptions
31	DTFSVPROT	Dead-time falling edge protection for value and sign This bit-field specifies the write protection for dead-time falling edge (value and sign). 0: Protect disable. DTFS and DTFCFG[15:0] are writable. 1: Protect enable. DTFS and DTFCFG[15:0] are read-only. **Note:** (1) The bit-field DTFCFG[15:9] is in SHRTIMER_STxACTL register. (2) This bit is not preloaded
30	DTFSPROT	Dead-time falling edge protection for sign This bit-field specifies the write protection for dead-time falling edge (only sign). 0: protect disable. DTFS in SHRTIMER_STxDTCTL register is writable. 1: protect enable. DTFS in SHRTIMER_STxDTCTL register is read-only. **Note:** This bit is not preloaded
29:26	Reserved	Must be kept at reset value
25	DTFS	The sign of falling edge dead-time value 0: The sign of falling edge dead-time value is positive. 1: The sign of falling edge dead-time value is negative. **Note:** This bit cannot be modified when DTFSPROT or DTFSVPROT bit in SHRTIMER_STxDTCTL register is set.
24:16	DTFCFG[8:0]	Falling edge dead-time value This bit-field controls the dead-time value of the following a falling edge of output prepare signal (OyPRE,y=0,1). DTFvalue = DTFCFG[15:0]x $t_{SHRTIMER\_DTGCK}$, $t_{SHRTIMER\_DTGCK}$ = 1/ $f_{SHRTIMER\_DTGCK}$. Writing this bit-field can change the low 9-bits of DTFCFG[15:0]. **Note:** (1) The bit-field DTFCFG[15:9] is in SHRTIMER_STxACTL register. (2) This bit-field cannot be modified when DTFSVPROT bit in SHRTIMER_STxDTCTL register is set.
15	DTRSVPROT	Dead-time rising edge protection for value and sign This bit-field specifies the write protection for dead-time rising edge (value and sign). 0: Protect disable. DTRS and DTRCFG[15:0] register are writable. 1: Protect enable. DTRS and DTRCFG[15:0] are read-only. **Note:** (1) The bit-field DTRCFG[15:9] is in SHRTIMER_STxACTL register. (2) This bit is not preloaded

14	DTRSPROT	Dead-time rising edge protection for sign This bit-field specifies the write protection for dead-time rising edge (only sign). 0: protect disable. DTRS in SHRTIMER_STxDTCTL register is writable. 1: protect enable. DTRS in SHRTIMER_STxDTCTL register is read-only. **Note:** This bit is not preloaded
13:10	DTGCKDIV[3:0]	Dead time generator clock division This bit-field can be configured by software to specify division ratio between the SHRTIMER clock (SHRTIMER_CK) and the dead-time generator clock (SHRTIMER_DTGCK). When DTGCKDIV[3] is '0', $f_{SHRTIMER\_DTGCK} = (8*f_{SHRTIMER\_CK})/2^{DTGCKDIV[2:0]}$. When DTGCKDIV[3] is '1', $f_{SHRTIMER\_DTGCK} = 2^{(DTGCKDIV[2:0]+4)}*f_{SHRTIMER\_CK}$ 0000: $f_{SHRTIMER\_DTGCK} = 8*f_{SHRTIMER\_CK}$ 0001: $f_{SHRTIMER\_DTGCK} = (8*f_{SHRTIMER\_CK})/2$ 0010: $f_{SHRTIMER\_DTGCK} = (8*f_{SHRTIMER\_CK})/4$ 0011: $f_{SHRTIMER\_DTGCK} = (8*f_{SHRTIMER\_CK})/8$ 0100: $f_{SHRTIMER\_DTGCK} = (8*f_{SHRTIMER\_CK})/16$ 0101: $f_{SHRTIMER\_DTGCK} = (8*f_{SHRTIMER\_CK})/32$ 0110: $f_{SHRTIMER\_DTGCK} = (8*f_{SHRTIMER\_CK})/64$ 0111: $f_{SHRTIMER\_DTGCK} = (8*f_{SHRTIMER\_CK})/128$ 1000: $f_{SHRTIMER\_DTGCK} = 16* f_{SHRTIMER\_CK}$ 1001: $f_{SHRTIMER\_DTGCK} = 32* f_{SHRTIMER\_CK}$ 1010: $f_{SHRTIMER\_DTGCK} = 64* f_{SHRTIMER\_CK}$ Other values are reserved. **Note:** This bit-field cannot be modified when any of the protect bits is set (DTFSPROT , DTFSVPROT, DTRSPROT and DTRSVPROT).
9	DTRS	The sign of rising edge dead-time value 0: The sign of rising edge dead-time value is positive. 1: The sign of rising edge dead-time value is negative. **Note:** This bit cannot be modified when DTRSPROT or DTRSVPROT bit in SHRTIMER_STxDTCTL register is set.
8:0	DTRCFG[8:0]	Rising edge dead-time value This bit-field controls the dead-time value of the following a rising edge of output prepare signal (OyPRE,y=0,1). DTRvalue = DTRCFG[15:0]x $t_{SHRTIMER\_DTGCK}$, $t_{SHRTIMER\_DTGCK} = 1/ f_{SHRTIMER\_DTGCK}$. Writing this bit-field can change the low 9-bits of DTRCFG[15:0]. **Note:** (1) The bit-field DTRCFG[15:9] is in SHRTIMER_STxACTL register. (2) This bit-field cannot be modified when DTRSVPROT bit in SHRTIMER_STxDTCTL register is set.

SHRTIMER Slave_TIMERx channel 0 set request register (SHRTIMER_STxCH0SET)

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
CH0SUP	CH0SEXEV9	CH0SEXEV8	CH0SEXEV7	CH0SEXEV6	CH0SEXEV5	CH0SEXEV4	CH0SEXEV3	CH0SEXEV2	CH0SEXEV1	CH0SEXEV0	CH0SSTEV8	CH0SSTEV7	CH0SSTEV6	CH0SSTEV5	CH0SSTEV4
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
CH0SSTEV3	CH0SSTEV2	CH0SSTEV1	CH0SSTEV0	CH0SMTCMP3	CH0SMTCMP2	CH0SMTCMP1	CH0SMTCMP0	CH0SMTPER	CH0SCMP3	CH0SCMP2	CH0SCMP1	CH0SCMP0	CH0SPER	CH0SRST	CH0SSEV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	CH0SUP	Update event generates channel 0 “set request” When this bit is set, update event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
30	CH0SEXEV9	External event 9 generates channel 0 “set request” Refer to CH0SEXEV0 description.
29	CH0SEXEV8	External event 8 generates channel 0 “set request” Refer to CH0SEXEV0 description.
28	CH0SEXEV7	External event 7 generates channel 0 “set request” Refer to CH0SEXEV0 description.
27	CH0SEXEV6	External event 6 generates channel 0 “set request” Refer to CH0SEXEV0 description.
26	CH0SEXEV5	External event 5 generates channel 0 “set request” Refer to CH0SEXEV0 description.
25	CH0SEXEV4	External event 4 generates channel 0 “set request” Refer to CH0SEXEV0 description.
24	CH0SEXEV3	External event 3 generates channel 0 “set request” Refer to CH0SEXEV0 description.
23	CH0SEXEV2	External event 2 generates channel 0 “set request” Refer to CH0SEXEV0 description.
22	CH0SEXEV1	External event 1 generates channel 0 “set request”

		Refer to CH0SEXEV0 description.
21	CH0SEXEV0	External event 0 generates channel 0 “set request” When this bit is set, external event 0 can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
20	CH0SSTEV8	Slave_TIMERx interconnection event 8 generates channel 0 “set request” Refer to CH0SSTEV0 description.
19	CH0SSTEV7	Slave_TIMERx interconnection event 7 generates channel 0 “set request” Refer to CH0SSTEV0 description.
18	CH0SSTEV6	Slave_TIMERx interconnection event 6 generates channel 0 “set request” Refer to CH0SSTEV0 description.
17	CH0SSTEV5	Slave_TIMERx interconnection event 5 generates channel 0 “set request” Refer to CH0SSTEV0 description.
16	CH0SSTEV4	Slave_TIMERx interconnection event 4 generates channel 0 “set request” Refer to CH0SSTEV0 description.
15	CH0SSTEV3	Slave_TIMERx interconnection event 3 generates channel 0 “set request” Refer to CH0SSTEV0 description.
14	CH0SSTEV2	Slave_TIMERx interconnection event 2 generates channel 0 “set request” Refer to CH0SSTEV0 description.
13	CH0SSTEV1	Slave_TIMERx interconnection event 1 generates channel 0 “set request” Refer to CH0SSTEV0 description.
12	CH0SSTEV0	Slave_TIMERx interconnection event 0 generates channel 0 “set request” When this bit is set, Slave_TIMERx interconnection event 0 can generate “set request”. Refer to ***Table 19-5. Slave_TIMER interconnection event.*** 0: The event cannot generate “set request”. 1: The event can generate “set request”.
11	CH0SMTCMP3	Master_TIMER compare 3 event generates channel 0 “set request” When this bit is set, Master_TIMER compare 3 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
10	CH0SMTCMP2	Master_TIMER compare 2 event generates channel 0 “set request” When this bit is set, Master_TIMER compare 2 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
9	CH0SMTCMP1	Master_TIMER compare 1 event generates channel 0 “set request” When this bit is set, Master_TIMER compare 1 event can generate “set request”. 0: The event cannot generate “set request”.

		1: The event can generate “set request”.
8	CH0SMTCMP0	Master_TIMER compare 0 event generates channel 0 “set request” When this bit is set, Master_TIMER compare 0 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
7	CH0SMTPER	Master_TIMER period event generates channel 0 “set request” In continuous mode, the Master_TIMER counter roll-over event can generate “set request”. In single pulse mode, the Master_TIMER reset event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generates “set request”.
6	CH0SCMP3	Slave_TIMERx compare 3 event generates channel 0 “set request” When this bit is set, Slave_TIMERx compare 3 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
5	CH0SCMP2	Slave_TIMERx compare 2 event generates channel 0 “set request” When this bit is set, Slave_TIMERx compare 2 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
4	CH0SCMP1	Slave_TIMERx compare 1 event generates channel 0 “set request” When this bit is set, Slave_TIMERx compare 1 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
3	CH0SCMP0	Slave_TIMERx compare 0 event generates channel 0 “set request” When this bit is set, Slave_TIMERx compare 0 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
2	CH0SPER	Slave_TIMERx period event generates channel 0 “set request” When this bit is set, Slave_TIMERx period event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
1	CH0SRST	Slave_TIMERx reset event generates channel 0 “set request” When this bit is set, Slave_TIMERx reset event from synchronous input and software can generate channel 0 “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”. **Note:** When this bit is set, the reset of other timers does not affect the output.
0	CH0SSEV	Software event generates channel 0 “set request” This bit is set by software and cleared by hardware automatically. When this bit is

set, it can generate channel 0 “set request”.

0: The event cannot generate “set request”.

1: The event can generate “set request”.

Note: This bit is not preloaded

SHRTIMER Slave_TIMERx channel 0 reset request register (SHRTIMER_STxCH0RST)

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
CH0RSUP	CH0RSEXEV9	CH0RSEXEV8	CH0RSEXEV7	CH0RSEXEV6	CH0RSEXEV5	CH0RSEXEV4	CH0RSEXEV3	CH0RSEXEV2	CH0RSEXEV1	CH0RSEXEV0	CH0RSSTEV8	CH0RSSTEV7	CH0RSSTEV6	CH0RSSTEV5	CH0RSSTEV4
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
CH0RSSTEV3	CH0RSSTEV2	CH0RSSTEV1	CH0RSSTEV0	CH0RSMTCMP3	CH0RSMTCMP2	CH0RSMTCMP1	CH0RSMTCMP0	CH0RSMTPER	CH0RSCMP3	CH0RSCMP2	CH0RSCMP1	CH0RSCMP0	CH0RSPER	CH0RSRST	CH0RSSEV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	CH0RSUP	Update event generates channel 0 “reset request” When this bit is set, update event can generate “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
30	CH0RSEXEV9	External event 9 generates channel 0 “reset request” Refer to CH0RSEXEV0 description.
29	CH0RSEXEV8	External event 8 generates channel 0 “reset request” Refer to CH0RSEXEV0 description.
28	CH0RSEXEV7	External event 7 generates channel 0 “reset request” Refer to CH0RSEXEV0 description.
27	CH0RSEXEV6	External event 6 generates channel 0 “reset request” Refer to CH0RSEXEV0 description.
26	CH0RSEXEV5	External event 5 generates channel 0 “reset request” Refer to CH0RSEXEV0 description.
25	CH0RSEXEV4	External event 4 generates channel 0 “reset request” Refer to CH0RSEXEV0 description.

24	CH0RSEXEV3	External event 3 generates channel 0 “reset request” Refer to CH0RSEXEV0 description.
23	CH0RSEXEV2	External event 2 generates channel 0 “reset request” Refer to CH0RSEXEV0 description.
22	CH0RSEXEV1	External event 1 generates channel 0 “reset request” Refer to CH0RSEXEV0 description.
21	CH0RSEXEV0	External event 0 generates channel 0 “reset request” When this bit is set, external event 0 can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
20	CH0RSSTEV8	Slave_TIMERx interconnection event 8 generates channel 0 “reset request” Refer to CH0RSSTEV0 description.
19	CH0RSSTEV7	Slave_TIMERx interconnection event 7 generates channel 0 “reset request” Refer to CH0RSSTEV0 description.
18	CH0RSSTEV6	Slave_TIMERx interconnection event 6 generates channel 0 “reset request” Refer to CH0RSSTEV0 description.
17	CH0RSSTEV5	Slave_TIMERx interconnection event 5 generates channel 0 “reset request” Refer to CH0RSSTEV0 description.
16	CH0RSSTEV4	Slave_TIMERx interconnection event 4 generates channel 0 “reset request” Refer to CH0RSSTEV0 description.
15	CH0RSSTEV3	Slave_TIMERx interconnection event 3 generates channel 0 “reset request” Refer to CH0RSSTEV0 description.
14	CH0RSSTEV2	Slave_TIMERx interconnection event 2 generates channel 0 “reset request” Refer to CH0RSSTEV0 description.
13	CH0RSSTEV1	Slave_TIMERx interconnection event 1 generates channel 0 “reset request” Refer to CH0RSSTEV0 description.
12	CH0RSSTEV0	Slave_TIMERx interconnection event 0 generates channel 0 “reset request” When this bit is set, Slave_TIMERx interconnection event 0 can generate channel “reset request”. Refer to ***Table 19-5. Slave_TIMER interconnection event.*** 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
11	CH0RSMTCMP3	Master_TIMER compare 3 event generates channel 0 “reset request” When this bit is set, Master_TIMER compare 3 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
10	CH0RSMTCMP2	Master_TIMER compare 2 event generates channel 0 “reset request”

		When this bit is set, Master_TIMER compare 2 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
9	CH0RSMTCMP1	Master_TIMER compare 1 event generates channel 0 “reset request” When this bit is set, Master_TIMER compare 1 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
8	CH0RSMTCMP0	Master_TIMER compare 0 event generates channel 0 “reset request” When this bit is set, Master_TIMER compare 0 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
7	CH0RSMTPER	Master_TIMER period event generates channel 0 “reset request” In continuous mode, the Master_TIMER counter roll-over event can generate channel “reset request”. In single pulse mode, the Master_TIMER reset event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
6	CH0RSCMP3	Slave_TIMERx compare 3 event generates channel 0 “reset request” When this bit is set, Slave_TIMERx compare 3 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
5	CH0RSCMP2	Slave_TIMERx compare 2 event generates channel 0 “reset request” When this bit is set, Slave_TIMERx compare 2 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
4	CH0RSCMP1	Slave_TIMERx compare 1 event generates channel 0 “reset request” When this bit is set, Slave_TIMERx compare 1 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
3	CH0RSCMP0	Slave_TIMERx compare 0 event generates channel 0 “reset request” When this bit is set, Slave_TIMERx compare 0 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.

2	CH0RSPER	Slave_TIMERx period event generates channel 0 “reset request” When this bit is set, Slave_TIMERx period event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
1	CH0RSRST	Slave_TIMERx reset event generates channel 0 “reset request” When this bit is set, Slave_TIMERx reset event from synchronous input and software can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”. **Note:** When this bit is set, the reset of other timers does not affect the output.
0	CH0RSSEV	Software event generates channel 0 “reset request” This bit is set by software and cleared by hardware automatically. When this bit is set, it will generate “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”. **Note:** This bit is not preloaded

SHRTIMER Slave_TIMERx channel 1 set request register (SHRTIMER_STxCH1SET)

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
CH1SUP	CH1SEXEV9	CH1SEXEV8	CH1SEXEV7	CH1SEXEV6	CH1SEXEV5	CH1SEXEV4	CH1SEXEV3	CH1SEXEV2	CH1SEXEV1	CH1SEXEV0	CH1SSTEV8	CH1SSTEV7	CH1SSTEV6	CH1SSTEV5	CH1SSTEV4
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
CH1SSTEV3	CH1SSTEV2	CH1SSTEV1	CH1SSTEV0	CH1SMTCMP3	CH1SMTCMP2	CH1SMTCMP1	CH1SMTCMP0	CH1SMTPER	CH1SCMP3	CH1SCMP2	CH1SCMP1	CH1SCMP0	CH1SPER	CH1SRST	CH1SSEV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	CH1SUP	Update event generates channel 1 “set request” When this bit is set, update event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
30	CH1SEXEV9	External event 9 generates channel 1 “set request”

		Refer to CH1SEXEV0 description.
29	CH1SEXEV8	External event 8 generates channel 1 “set request” Refer to CH1SEXEV0 description.
28	CH1SEXEV7	External event 7 generates channel 1 “set request” Refer to CH1SEXEV0 description.
27	CH1SEXEV6	External event 6 generates channel 1 “set request” Refer to CH1SEXEV0 description.
26	CH1SEXEV5	External event 5 generates channel 1 “set request” Refer to CH1SEXEV0 description.
25	CH1SEXEV4	External event 4 generates channel 1 “set request” Refer to CH1SEXEV0 description.
24	CH1SEXEV3	External event 3 generates channel 1 “set request” Refer to CH1SEXEV0 description.
23	CH1SEXEV2	External event 2 generates channel 1 “set request” Refer to CH1SEXEV0 description.
22	CH1SEXEV1	External event 1 generates channel 1 “set request” Refer to CH1SEXEV0 description.
21	CH1SEXEV0	External event 0 generates channel 1 “set request” When this bit is set, external event 0 can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
20	CH1SSTEV8	Slave_TIMERx interconnection event 8 generates channel 1 “set request” Refer to CH1SSTEV0 description.
19	CH1SSTEV7	Slave_TIMERx interconnection event 7 generates channel 1 “set request” Refer to CH1SSTEV0 description.
18	CH1SSTEV6	Slave_TIMERx interconnection event 6 generates channel 1 “set request” Refer to CH1SSTEV0 description.
17	CH1SSTEV5	Slave_TIMERx interconnection event 5 generates channel 1 “set request” Refer to CH1SSTEV0 description.
16	CH1SSTEV4	Slave_TIMERx interconnection event 4 generates channel 1 “set request” Refer to CH1SSTEV0 description.
15	CH1SSTEV3	Slave_TIMERx interconnection event 3 generates channel 1 “set request” Refer to CH1SSTEV0 description.
14	CH1SSTEV2	Slave_TIMERx interconnection event 2 generates channel 1 “set request” Refer to CH1SSTEV0 description.

13	CH1SSTEV1	Slave_TIMERx interconnection event 1 generates channel 1 “set request” Refer to CH1SSTEV0 description.
12	CH1SSTEV0	Slave_TIMERx interconnection event 0 generates channel 1 “set request” When this bit is set, Slave_TIMERx interconnection event 0 can generate “set request”. Refer to ***Table 19-5. Slave_TIMER interconnection event.*** 0: The event cannot generate “set request”. 1: The event can generate “set request”.
11	CH1SMTCMP3	Master_TIMER compare 3 event generates channel 1 “set request” When this bit is set, Master_TIMER compare 3 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
10	CH1SMTCMP2	Master_TIMER compare 2 event generates channel 1 “set request” When this bit is set, Master_TIMER compare 2 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
9	CH1SMTCMP1	Master_TIMER compare 1 event generates channel 1 “set request” When this bit is set, Master_TIMER compare 1 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
8	CH1SMTCMP0	Master_TIMER compare 0 event generates channel 1 “set request” When this bit is set, Master_TIMER compare 0 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
7	CH1SMTPER	Master_TIMER period event generates channel 1 “set request” In continuous mode, the Master_TIMER counter roll-over event can generate “set request”. In single pulse mode, the Master_TIMER reset event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generates “set request”.
6	CH1SCMP3	Slave_TIMERx compare 3 event generates channel 1 “set request” When this bit is set, Slave_TIMERx compare 3 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
5	CH1SCMP2	Slave_TIMERx compare 2 event generates channel 1 “set request” When this bit is set, Slave_TIMERx compare 2 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
4	CH1SCMP1	Slave_TIMERx compare 1 event generates channel 1 “set request” When this bit is set, Slave_TIMERx compare 1 event can generate “set request”.

0: The event cannot generate “set request”.
1: The event can generate “set request”.

3	CH1SCMP0	Slave_TIMERx compare 0 event generates channel 1 “set request” When this bit is set, Slave_TIMERx compare 0 event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
2	CH1SPER	Slave_TIMERx period event generates channel 1 “set request” When this bit is set, Slave_TIMERx period event can generate “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”.
1	CH1SRST	Slave_TIMERx reset event generates channel 1 “set request” When this bit is set, Slave_TIMERx reset event from synchronous input and software can generate channel 1 “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”. **Note:** When this bit is set, the reset of other timers does not affect the output.
0	CH1SSEV	Software event generates channel 1 “set request” This bit is set by software and cleared by hardware automatically. When this bit is set, it can generate channel 1 “set request”. 0: The event cannot generate “set request”. 1: The event can generate “set request”. **Note:** This bit is not preloaded

SHRTIMER Slave_TIMERx channel 1 reset request register (SHRTIMER_STxCH1RST)

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
CH1RSUP	CH1RSEXEV9	CH1RSEXEV8	CH1RSEXEV7	CH1RSEXEV6	CH1RSEXEV5	CH1RSEXEV4	CH1RSEXEV3	CH1RSEXEV2	CH1RSEXEV1	CH1RSEXEV0	CH1RSSTEV8	CH1RSSTEV7	CH1RSSTEV6	CH1RSSTEV5	CH1RSSTEV4
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
CH1RSSTEV3	CH1RSSTEV2	CH1RSSTEV1	CH1RSSTEV0	CH1RSMTCMP3	CH1RSMTCMP2	CH1RSMTCMP1	CH1RSMTCMP0	CH1RSMTPER	CH1RSCMP3	CH1RSCMP2	CH1RSCMP1	CH1RSCMP0	CH1RSPER	CH1RSRST	CH1RSSEV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions

31	CH1RSUP	Update event generates channel 1 “reset request” When this bit is set, update event can generate “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
30	CH1RSEXEV9	External event 9 generates channel 1 “reset request” Refer to CH1RSEXEV0 description.
29	CH1RSEXEV8	External event 8 generates channel 1 “reset request” Refer to CH1RSEXEV0 description.
28	CH1RSEXEV7	External event 7 generates channel 1 “reset request” Refer to CH1RSEXEV0 description.
27	CH1RSEXEV6	External event 6 generates channel 1 “reset request” Refer to CH1RSEXEV0 description.
26	CH1RSEXEV5	External event 5 generates channel 1 “reset request” Refer to CH1RSEXEV0 description.
25	CH1RSEXEV4	External event 4 generates channel 1 “reset request” Refer to CH1RSEXEV0 description.
24	CH1RSEXEV3	External event 3 generates channel 1 “reset request” Refer to CH1RSEXEV0 description.
23	CH1RSEXEV2	External event 2 generates channel 1 “reset request” Refer to CH1RSEXEV0 description.
22	CH1RSEXEV1	External event 1 generates channel 1 “reset request” Refer to CH1RSEXEV0 description.
21	CH1RSEXEV0	External event 0 generates channel 1 “reset request” When this bit is set, external event 0 can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
20	CH1RSSTEV8	Slave_TIMERx interconnection event 8 generates channel 1 “reset request” Refer to CH1RSSTEV0 description.
19	CH1RSSTEV7	Slave_TIMERx interconnection event 7 generates channel 1 “reset request” Refer to CH1RSSTEV0 description.
18	CH1RSSTEV6	Slave_TIMERx interconnection event 6 generates channel 1 “reset request” Refer to CH1RSSTEV0 description.
17	CH1RSSTEV5	Slave_TIMERx interconnection event 5 generates channel 1 “reset request” Refer to CH1RSSTEV0 description.
16	CH1RSSTEV4	Slave_TIMERx interconnection event 4 generates channel 1 “reset request” Refer to CH1RSSTEV0 description.

15	CH1RSSTEV3	Slave_TIMERx interconnection event 3 generates channel 1 “reset request” Refer to CH1RSSTEV0 description.
14	CH1RSSTEV2	Slave_TIMERx interconnection event 2 generates channel 1 “reset request” Refer to CH1RSSTEV0 description.
13	CH1RSSTEV1	Slave_TIMERx interconnection event 1 generates channel 1 “reset request” Refer to CH1RSSTEV0 description.
12	CH1RSSTEV0	Slave_TIMERx interconnection event 0 generates channel 1 “reset request” When this bit is set, Slave_TIMERx interconnection event 0 can generate channel “reset request”. Refer to ***Table 19-5. Slave_TIMER interconnection event.*** 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
11	CH1RSMTCMP3	Master_TIMER compare 3 event generates channel 1 “reset request” When this bit is set, Master_TIMER compare 3 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
10	CH1RSMTCMP2	Master_TIMER compare 2 event generates channel 1 “reset request” When this bit is set, Master_TIMER compare 2 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
9	CH1RSMTCMP1	Master_TIMER compare 1 event generates channel 1 “reset request” When this bit is set, Master_TIMER compare 1 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
8	CH1RSMTCMP0	Master_TIMER compare 0 event generates channel 1 “reset request” When this bit is set, Master_TIMER compare 0 event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
7	CH1RSMTPER	Master_TIMER period event generates channel 1 “reset request” In continuous mode, the Master_TIMER counter roll-over event can generate channel “reset request”. In single pulse mode, the Master_TIMER reset event can generate channel “reset request”. 0: The event cannot generate “reset request”. 1: The event can generate “reset request”.
6	CH1RSCMP3	Slave_TIMERx compare 3 event generates channel 1 “reset request” When this bit is set, Slave_TIMERx compare 3 event can generate channel “reset

request".
0: The event cannot generate "reset request".
1: The event can generate "reset request".

5 CH1RSCMP2 Slave_TIMERx compare 2 event generates channel 1 "reset request"
When this bit is set, Slave_TIMERx compare 2 event can generate channel "reset request".
0: The event cannot generate "reset request".
1: The event can generate "reset request".

4 CH1RSCMP1 Slave_TIMERx compare 1 event generates channel 1 "reset request"
When this bit is set, Slave_TIMERx compare 1 event can generate channel "reset request".
0: The event cannot generate "reset request".
1: The event can generate "reset request".

3 CH1RSCMP0 Slave_TIMERx compare 0 event generates channel 1 "reset request"
When this bit is set, Slave_TIMERx compare 0 event can generate channel "reset request".
0: The event cannot generate "reset request".
1: The event can generate "reset request".

2 CH1RSPER Slave_TIMERx period event generates channel 1 "reset request"
When this bit is set, Slave_TIMERx period event can generate channel "reset request".
0: The event cannot generate "reset request".
1: The event can generate "reset request".

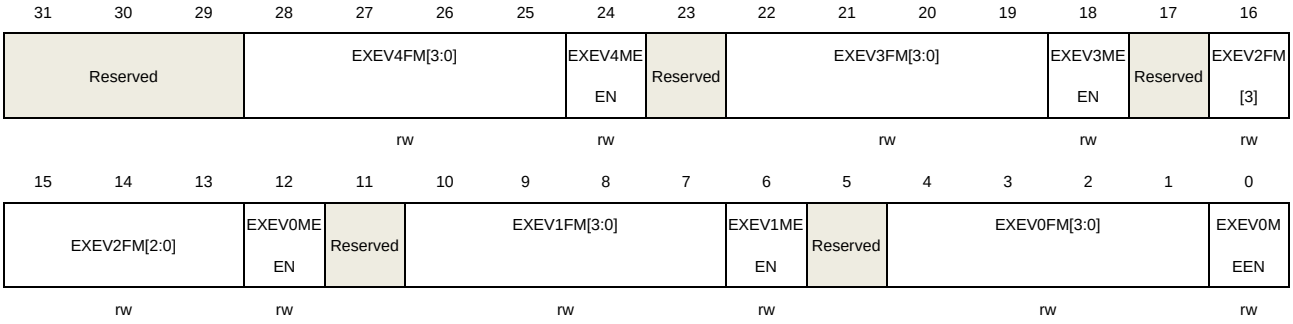
1 CH1RSRST Slave_TIMERx reset event generates channel 1 "reset request"
When this bit is set, Slave_TIMERx reset event from synchronous input and software can generate channel "reset request".
0: The event cannot generate "reset request".
1: The event can generate "reset request".
Note: When this bit is set, the reset of other timers does not affect the output.

0 CH1RSSEV Software event generates channel 1 "reset request"
This bit is set by software and cleared by hardware automatically. When this bit is set, it will generate "reset request".
0: The event cannot generate "reset request".
1: The event can generate "reset request".
Note: This bit is not preloaded

SHRTIMER Slave_TIMERx external event filter configuration register 0 (SHRTIMER_STxEXEVFCFG0)

Address offset: 0x4C
Reset value: 0x0000 0000

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



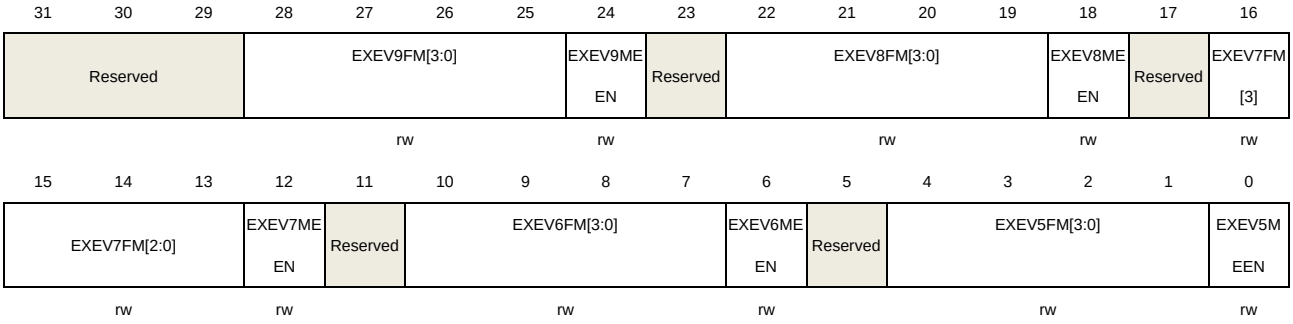
Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28:25	EXEV4FM[3:0]	External event 4 filter mode Refer to EXEV0FM[3:0] description.
24	EXEV4MEEN	External event 4 memorized enable Refer to EXEV0MEEN description.
23	Reserved	Must be kept at reset value
22:19	EXEV3FM[3:0]	External event 3 filter mode Refer to EXEV0FM[3:0] description.
18	EXEV3MEEN	External event 3 memorized enable Refer to EXEV0MEEN description.
17	Reserved	Must be kept at reset value
16:13	EXEV2FM[3:0]	External event 2 filter mode Refer to EXEV0FM[3:0] description.
12	EXEV2MEEN	External event 2 memorized enable Refer to EXEV0MEEN description.
11	Reserved	Must be kept at reset value
10:7	EXEV1FM[3:0]	External event 1 filter mode Refer to EXEV0FM[3:0] description.
6	EXEV1MEEN	External event 1 memorized enable Refer to EXEV0MEEN description.

5	Reserved	Must be kept at reset value
4:1	EXEV0FM[3:0]	External event 0 filter mode In blanking mode, the external event is ignored if it occurs during a blank. In windowing mode, the external event is taken into account only if it occurs within a given time window. 0000: filter mode disable. 0001: Blanking mode. The blank is from counter reset to SHRTIMER_STxCMP0V. 0010: Blanking mode. The blank is from counter reset to SHRTIMER_STxCMP1V. 0011: Blanking mode. The blank is from counter reset to SHRTIMER_STxCMP2V. 0100: Blanking mode. The blank is from counter reset to SHRTIMER_STxCMP3V. 0101: Blanking mode. The blank is from other Slave_TIMERy(not Slave_TIMERx): STBLKSRC0 0110: Blanking mode. The blank is from other Slave_TIMERy(not Slave_TIMERx): STBLKSRC1 0111: Blanking mode. The blank is from other Slave_TIMERy(not Slave_TIMERx): STBLKSRC2 1000: Blanking mode. The blank is from other Slave_TIMERy(not Slave_TIMERx): STBLKSRC3 1001: Blanking mode. The blank is from other Slave_TIMERy(not Slave_TIMERx): STBLKSRC4 1010: Blanking mode. The blank is from other Slave_TIMERy(not Slave_TIMERx): STBLKSRC5 1011: Blanking mode. The blank is from other Slave_TIMERy(not Slave_TIMERx): STBLKSRC6 1100: Blanking mode. The blank is from other Slave_TIMERy(not Slave_TIMERx): STBLKSRC7 1101: Windowing mode. The windowing is from counter reset to SHRTIMER_STxCMP1V. 1110: Windowing mode. The windowing is from counter reset to SHRTIMER_STxCMP2V. 1111: Windowing mode. The windowing is from other Slave_TIMERy(not Slave_TIMERx):STWDSRC **Note:** (1) This bit-field must not be modified once the counter is enabled (STxCEN bit set) (2) The value of the compare register which used for filter must be above 0.
0	EXEV0MEEN	External event 0 memory 0: External event memory disable. 1: External event memory enable. The memorized event is generated as soon as the blanking period or windowing period is completed. **Note:** (1) This bit-field must not be modified once the counter is enabled (STxCEN bit set) (2) When this bit is set, a timeout event can be generated in window mode.

SHRTIMER Slave_TIMERx external event filter configuration register 1 (SHRTIMER_STxEXEVFCFG1)

Address offset: 0x50
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28:25	EXEV9FM[3:0]	External event 9 filter mode Refer to EXEV0FM[3:0] in SHRTIMER_STxEXEVFCFG0 description.
24	EXEV9MEEN	External event 9 memorized enable Refer to EXEV0MEEN in SHRTIMER_STxEXEVFCFG0 description.
23	Reserved	Must be kept at reset value
22:19	EXEV8FM[3:0]	External event 8 filter mode Refer to EXEV0FM[3:0] in SHRTIMER_STxEXEVFCFG0 description.
18	EXEV8MEEN	External event 8 memorized enable Refer to EXEV0MEEN in SHRTIMER_STxEXEVFCFG0 description.
17	Reserved	Must be kept at reset value
16:13	EXEV7FM[3:0]	External event 7 filter mode Refer to EXEV0FM[3:0] in SHRTIMER_STxEXEVFCFG0 description.
12	EXEV7MEEN	External event 7 memorized enable Refer to EXEV0MEEN in SHRTIMER_STxEXEVFCFG0 description.
11	Reserved	Must be kept at reset value
10:7	EXEV6FM[3:0]	External event 6 filter mode Refer to EXEV0FM[3:0] in SHRTIMER_STxEXEVFCFG0 description.
6	EXEV6MEEN	External event 6 memorized enable Refer to EXEV0MEEN in SHRTIMER_STxEXEVFCFG0 description.

5	Reserved	Must be kept at reset value
4:1	EXEV5FM[3:0]	External event 5 filter mode Refer to EXEV0FM[3:0] in SHRTIMER_STxEXEVFCFG0 description.
0	EXEV5MEEN	External event 0 memorized enable Refer to EXEV0MEEN in SHRTIMER_STxEXEVFCFG0 description.

SHRTIMER Slave_TIMERx counter reset register (SHRTIMER_STxCNTRST)

Address offset: 0x54
Reset value: 0x0000 0000

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

For Slave_TIMER0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	ST4CMP3RST	ST4CMP1RST	ST4CMP0RST	ST3CMP3RST	ST3CMP1RST	ST3CMP0RST	ST2CMP3RST	ST2CMP1RST	ST2CMP0RST	ST1CMP3RST	ST1CMP1RST	ST1CMP0RST	EXEV9RST	EXEV8RST	EXEV7RST
	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
EXEV6RST	EXEV5RST	EXEV4RST	EXEV3RST	EXEV2RST	EXEV1RST	EXEV0RST	MTCMP3RST	MTCMP2RST	MTCMP1RST	MTCMP0RST	MTPERRST	CMP3RST	CMP1RST	UPRST	Reserved
rw	rw	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	ST4CMP3RST	Slave_TIMER4 compare 3 event resets counter Refer to ST4CMP0RST description.
29	ST4CMP1RST	Slave_TIMER4 compare 1 event resets counter Refer to ST4CMP0RST description.
28	ST4CMP0RST	Slave_TIMER4 compare 0 event resets counter This bit specifies whether the Slave_TIMER4 compare 0 event can reset the counter. 0: Slave_TIMER4 compare 0 event do not reset counter 1: Slave_TIMER4 compare 0 event resets counter
27	ST3CMP3RST	Slave_TIMER3 compare 3 event resets counter Refer to ST3CMP0RST description.
26	ST3CMP1RST	Slave_TIMER3 compare 1 event resets counter Refer to ST3CMP0RST description.
25	ST3CMP0RST	Slave_TIMER3 compare 0 event resets counter This bit specifies whether the Slave_TIMER3 compare 0 event can reset the

		counter. 0: Slave_TIMER3 compare 0 event do not reset counter 1: Slave_TIMER3 compare 0 event resets counter
24	ST2CMP3RST	Slave_TIMER2 compare 3 event resets counter Refer to ST2CMP0RST description.
23	ST2CMP1RST	Slave_TIMER2 compare 1 event resets counter Refer to ST2CMP0RST description.
22	ST2CMP0RST	Slave_TIMER2 compare 0 event resets counter This bit specifies whether the Slave_TIMER2 compare 0 event can reset the counter. 0: Slave_TIMER2 compare 0 event do not reset counter 1: Slave_TIMER2 compare 0 event resets counter
21	ST1CMP3RST	Slave_TIMER1 compare 3 event resets counter Refer to ST1CMP0RST description.
20	ST1CMP1RST	Slave_TIMER1 compare 1 event resets counter Refer to ST1CMP0RST description.
19	ST1CMP0RST	Slave_TIMER1 compare 0 event resets counter This bit specifies whether the Slave_TIMER1 compare 0 event can reset the counter. 0: Slave_TIMER1 compare 0 event do not reset counter 1: Slave_TIMER1 compare 0 event resets counter
18	EXEV9RST	External event 9 resets counter Refer to EXEV0RST description.
17	EXEV8RST	External event 8 resets counter Refer to EXEV0RST description.
16	EXEV7RST	External event 7 resets counter Refer to EXEV0RST description.
15	EXEV6RST	External event 6 resets counter Refer to EXEV0RST description.
14	EXEV5RST	External event 5 resets counter Refer to EXEV0RST description.
13	EXEV4RST	External event 4 resets counter Refer to EXEV0RST description.
12	EXEV3RST	External event 3 resets counter Refer to EXEV0RST description.
11	EXEV2RST	External event 2 resets counter

		Refer to EXEV0RST description.
10	EXEV1RST	External event 1 resets counter Refer to EXEV0RST description.
9	EXEV0RST	External event 0 resets counter This bit specifies whether the External event 0 can reset the counter. 0: External event 0 do not reset counter. 1: External event 0 resets counter.
8	MTCMP3RST	Master_TIMER compare 3 event resets counter Refer to MTCMP0RST description
7	MTCMP2RST	Master_TIMER compare 2 event resets counter Refer to MTCMP0RST description
6	MTCMP1RST	Master_TIMER compare 1 event resets counter Refer to MTCMP0RST description
5	MTCMP0RST	Master_TIMER compare 0 event resets counter This bit specifies whether the Master_TIMER compare 0 event can reset the counter. 0: Master_TIMER compare 0 event do not reset counter 1: Master_TIMER compare 0 event resets counter
4	MTPERRST	Master_TIMER period event resets counter This bit specifies whether the Master_TIMER period event can reset the counter. 0: Master_TIMER period event do not reset counter 1: Master_TIMER period event resets counter
3	CMP3RST	Slave_TIMER0 compare 3 event resets counter Refer to CMP1RST description
2	CMP1RST	Slave_TIMER0 compare 1 event resets counter This bit specifies whether the compare 1 event can reset the counter. 0: Compare 1 event do not reset counter 1: Compare 1 event resets counter
1	UPRST	Slave_TIMER0 update event resets counter This bit specifies whether the update event can reset the counter. 0: Update event do not reset counter 1: Update event resets counter
0	Reserved	Must be kept at reset value

For Slave_TIMER1

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

Reserved	ST4CMP3RST	ST4CMP1RST	ST4CMP0RST	ST3CMP3RST	ST3CMP1RST	ST3CMP0RST	ST2CMP3RST	ST2CMP1RST	ST2CMP0RST	ST0CMP3RST	ST0CMP1RST	ST0CMP0RST	EXEV9RST	EXEV8RST	EXEV7RST
	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

EXEV6RST	EXEV5RST	EXEV4RST	EXEV3RST	EXEV2RST	EXEV1RST	EXEV0RST	MTCMP3RST	MTCMP2RST	MTCMP1RST	MTCMP0RST	MTPERRST	CMP3RST	CMP1RST	UPRST	Reserved
rw	rw	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	ST4CMP3RST	Slave_TIMER4 compare 3 event resets counter Refer to ST4CMP0RST description.
29	ST4CMP1RST	Slave_TIMER4 compare 1 event resets counter Refer to ST4CMP0RST description.
28	ST4CMP0RST	Slave_TIMER4 compare 0 event resets counter This bit specifies whether the Slave_TIMER4 compare 0 event can reset the counter. 0: Slave_TIMER4 compare 0 event do not reset counter 1: Slave_TIMER4 compare 0 event resets counter
27	ST3CMP3RST	Slave_TIMER3 compare 3 event resets counter Refer to ST3CMP0RST description.
26	ST3CMP1RST	Slave_TIMER3 compare 1 event resets counter Refer to ST3CMP0RST description.
25	ST3CMP0RST	Slave_TIMER3 compare 0 event resets counter This bit specifies whether the Slave_TIMER3 compare 0 event can reset the counter. 0: Slave_TIMER3 compare 0 event do not reset counter 1: Slave_TIMER3 compare 0 event resets counter
24	ST2CMP3RST	Slave_TIMER2 compare 3 event resets counter Refer to ST2CMP0RST description.
23	ST2CMP1RST	Slave_TIMER2 compare 1 event resets counter Refer to ST2CMP0RST description.
22	ST2CMP0RST	Slave_TIMER2 compare 0 event resets counter This bit specifies whether the Slave_TIMER2 compare 0 event can reset the counter. 0: Slave_TIMER2 compare 0 event do not reset counter 1: Slave_TIMER2 compare 0 event resets counter
21	ST0CMP3RST	Slave_TIMER0 compare 3 event resets counter

		Refer to ST0CMP0RST description.
20	ST0CMP1RST	Slave_TIMER0 compare 1 event resets counter Refer to ST0CMP0RST description.
19	ST0CMP0RST	Slave_TIMER0 compare 0 event resets counter This bit specifies whether the Slave_TIMER0 compare 0 event can reset the counter. 0: Slave_TIMER0 compare 0 event do not reset counter 1: Slave_TIMER0 compare 0 event resets counter
18	EXEV9RST	External event 9 resets counter Refer to EXEV0RST description.
17	EXEV8RST	External event 8 resets counter Refer to EXEV0RST description.
16	EXEV7RST	External event 7 resets counter Refer to EXEV0RST description.
15	EXEV6RST	External event 6 resets counter Refer to EXEV0RST description.
14	EXEV5RST	External event 5 resets counter Refer to EXEV0RST description.
13	EXEV4RST	External event 4 resets counter Refer to EXEV0RST description.
12	EXEV3RST	External event 3 resets counter Refer to EXEV0RST description.
11	EXEV2RST	External event 2 resets counter Refer to EXEV0RST description.
10	EXEV1RST	External event 1 resets counter Refer to EXEV0RST description.
9	EXEV0RST	External event 0 resets counter This bit specifies whether the External event 0 can reset the counter. 0: External event 0 do not reset counter. 1: External event 0 resets counter.
8	MTCMP3RST	Master_TIMER compare 3 event resets counter Refer to MTCMP0RST description
7	MTCMP2RST	Master_TIMER compare 2 event resets counter Refer to MTCMP0RST description
6	MTCMP1RST	Master_TIMER compare 1 event resets counter Refer to MTCMP0RST description

5	MTCMP0RST	Master_TIMER compare 0 event resets counter This bit specifies whether the Master_TIMER compare 0 event can reset the counter. 0: Master_TIMER compare 0 event do not reset counter 1: Master_TIMER compare 0 event resets counter
4	MTPERRST	Master_TIMER period event resets counter This bit specifies whether the Master_TIMER period event can reset the counter. 0: Master_TIMER period event do not reset counter 1: Master_TIMER period event resets counter
3	CMP3RST	Slave_TIMER1 compare 3 event resets counter Refer to CMP1RST description
2	CMP1RST	Slave_TIMER1 compare 1 event resets counter This bit specifies whether the compare 1 event can reset the counter. 0: Compare 1 event do not reset counter 1: Compare 1 event resets counter
1	UPRST	Slave_TIMER1 update event resets counter This bit specifies whether the update event can reset the counter. 0: Update event do not reset counter 1: Update event resets counter
0	Reserved	Must be kept at reset value

For Slave_TIMER2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	ST4CMP3RST	ST4CMP1RST	ST4CMP0RST	ST3CMP3RST	ST3CMP1RST	ST3CMP0RST	ST1CMP3RST	ST1CMP1RST	ST1CMP0RST	ST0CMP3RST	ST0CMP1RST	ST0CMP0RST	EXEV9RST	EXEV8RST	EXEV7RST
	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
EXEV6RST	EXEV5RST	EXEV4RST	EXEV3RST	EXEV2RST	EXEV1RST	EXEV0RST	MTCMP3RST	MTCMP2RST	MTCMP1RST	MTCMP0RST	MTPERRST	CMP3RST	CMP1RST	UPRST	Reserved
rw	rw	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	ST4CMP3RST	Slave_TIMER4 compare 3 event resets counter Refer to ST4CMP0RST description.
29	ST4CMP1RST	Slave_TIMER4 compare 1 event resets counter Refer to ST4CMP0RST description.
28	ST4CMP0RST	Slave_TIMER4 compare 0 event resets counter

		This bit specifies whether the Slave_TIMER4 compare 0 event can reset the counter. 0: Slave_TIMER4 compare 0 event do not reset counter 1: Slave_TIMER4 compare 0 event resets counter
27	ST3CMP3RST	Slave_TIMER3 compare 3 event resets counter Refer to ST3CMP0RST description.
26	ST3CMP1RST	Slave_TIMER3 compare 1 event resets counter Refer to ST3CMP0RST description.
25	ST3CMP0RST	Slave_TIMER3 compare 0 event resets counter This bit specifies whether the Slave_TIMER3 compare 0 event can reset the counter. 0: Slave_TIMER3 compare 0 event do not reset counter 1: Slave_TIMER3 compare 0 event resets counter
24	ST1CMP3RST	Slave_TIMER1 compare 3 event resets counter Refer to ST1CMP0RST description.
23	ST1CMP1RST	Slave_TIMER1 compare 1 event resets counter Refer to ST1CMP0RST description.
22	ST1CMP0RST	Slave_TIMER1 compare 0 event resets counter This bit specifies whether the Slave_TIMER1 compare 0 event can reset the counter. 0: Slave_TIMER1 compare 0 event do not reset counter 1: Slave_TIMER1 compare 0 event resets counter
21	ST0CMP3RST	Slave_TIMER0 compare 3 event resets counter Refer to ST0CMP0RST description.
20	ST0CMP1RST	Slave_TIMER0 compare 1 event resets counter Refer to ST0CMP0RST description.
19	ST0CMP0RST	Slave_TIMER0 compare 0 event resets counter This bit specifies whether the Slave_TIMER0 compare 0 event can reset the counter. 0: Slave_TIMER0 compare 0 event do not reset counter 1: Slave_TIMER0 compare 0 event resets counter
18	EXEV9RST	External event 9 resets counter Refer to EXEV0RST description.
17	EXEV8RST	External event 8 resets counter Refer to EXEV0RST description.
16	EXEV7RST	External event 7 resets counter Refer to EXEV0RST description.

15	EXEV6RST	External event 6 resets counter Refer to EXEV0RST description.
14	EXEV5RST	External event 5 resets counter Refer to EXEV0RST description.
13	EXEV4RST	External event 4 resets counter Refer to EXEV0RST description.
12	EXEV3RST	External event 3 resets counter Refer to EXEV0RST description.
11	EXEV2RST	External event 2 resets counter Refer to EXEV0RST description.
10	EXEV1RST	External event 1 resets counter Refer to EXEV0RST description.
9	EXEV0RST	External event 0 resets counter This bit specifies whether the External event 0 can reset the counter. 0: External event 0 do not reset counter. 1: External event 0 resets counter.
8	MTCMP3RST	Master_TIMER compare 3 event resets counter Refer to MTCMP0RST description
7	MTCMP2RST	Master_TIMER compare 2 event resets counter Refer to MTCMP0RST description
6	MTCMP1RST	Master_TIMER compare 1 event resets counter Refer to MTCMP0RST description
5	MTCMP0RST	Master_TIMER compare 0 event resets counter This bit specifies whether the Master_TIMER compare 0 event can reset the counter. 0: Master_TIMER compare 0 event do not reset counter 1: Master_TIMER compare 0 event resets counter
4	MTPERRST	Master_TIMER period event resets counter This bit specifies whether the Master_TIMER period event can reset the counter. 0: Master_TIMER period event do not reset counter 1: Master_TIMER period event resets counter
3	CMP3RST	Slave_TIMER2 compare 3 event resets counter Refer to CMP1RST description
2	CMP1RST	Slave_TIMER2 compare 1 event resets counter This bit specifies whether the compare 1 event can reset the counter. 0: Compare 1 event do not reset counter 1: Compare 1 event resets counter

1	UPRST	Slave_TIMER2 update event resets counter This bit specifies whether the update event can reset the counter. 0: Update event do not reset counter 1: Update event resets counter
0	Reserved	Must be kept at reset value

For Slave_TIMER3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	ST4CMP3RST	ST4CMP1RST	ST4CMP0RST	ST2CMP3RST	ST2CMP1RST	ST2CMP0RST	ST1CMP3RST	ST1CMP1RST	ST1CMP0RST	ST0CMP3RST	ST0CMP1RST	ST0CMP0RST	EXEV9RST	EXEV8RST	EXEV7RST
	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
EXEV6RST	EXEV5RST	EXEV4RST	EXEV3RST	EXEV2RST	EXEV1RST	EXEV0RST	MTCMP3RST	MTCMP2RST	MTCMP1RST	MTCMP0RST	MTPERRST	CMP3RST	CMP1RST	UPRST	Reserved
rw	rw	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	ST4CMP3RST	Slave_TIMER4 compare 3 event resets counter Refer to ST4CMP0RST description.
29	ST4CMP1RST	Slave_TIMER4 compare 1 event resets counter Refer to ST4CMP0RST description.
28	ST4CMP0RST	Slave_TIMER4 compare 0 event resets counter This bit specifies whether the Slave_TIMER4 compare 0 event can reset the counter. 0: Slave_TIMER4 compare 0 event do not reset counter 1: Slave_TIMER4 compare 0 event resets counter
27	ST2CMP3RST	Slave_TIMER2 compare 3 event resets counter Refer to ST2CMP0RST description.
26	ST2CMP1RST	Slave_TIMER2 compare 1 event resets counter Refer to ST2CMP0RST description.
25	ST2CMP0RST	Slave_TIMER2 compare 0 event resets counter This bit specifies whether the Slave_TIMER2 compare 0 event can reset the counter. 0: Slave_TIMER2 compare 0 event do not reset counter 1: Slave_TIMER2 compare 0 event resets counter
24	ST1CMP3RST	Slave_TIMER1 compare 3 event resets counter

		Refer to ST1CMP0RST description.
23	ST1CMP1RST	Slave_TIMER1 compare 1 event resets counter Refer to ST1CMP0RST description.
22	ST1CMP0RST	Slave_TIMER1 compare 0 event resets counter This bit specifies whether the Slave_TIMER1 compare 0 event can reset the counter. 0: Slave_TIMER1 compare 0 event do not reset counter 1: Slave_TIMER1 compare 0 event resets counter
21	ST0CMP3RST	Slave_TIMER0 compare 3 event resets counter Refer to ST0CMP0RST description.
20	ST0CMP1RST	Slave_TIMER0 compare 1 event resets counter Refer to ST0CMP0RST description.
19	ST0CMP0RST	Slave_TIMER0 compare 0 event resets counter This bit specifies whether the Slave_TIMER0 compare 0 event can reset the counter. 0: Slave_TIMER0 compare 0 event do not reset counter 1: Slave_TIMER0 compare 0 event resets counter
18	EXEV9RST	External event 9 resets counter Refer to EXEV0RST description.
17	EXEV8RST	External event 8 resets counter Refer to EXEV0RST description.
16	EXEV7RST	External event 7 resets counter Refer to EXEV0RST description.
15	EXEV6RST	External event 6 resets counter Refer to EXEV0RST description.
14	EXEV5RST	External event 5 resets counter Refer to EXEV0RST description.
13	EXEV4RST	External event 4 resets counter Refer to EXEV0RST description.
12	EXEV3RST	External event 3 resets counter Refer to EXEV0RST description.
11	EXEV2RST	External event 2 resets counter Refer to EXEV0RST description.
10	EXEV1RST	External event 1 resets counter Refer to EXEV0RST description.
9	EXEV0RST	External event 0 resets counter

		This bit specifies whether the External event 0 can reset the counter. 0: External event 0 do not reset counter. 1: External event 0 resets counter.
8	MTCMP3RST	Master_TIMER compare 3 event resets counter Refer to MTCMP0RST description
7	MTCMP2RST	Master_TIMER compare 2 event resets counter Refer to MTCMP0RST description
6	MTCMP1RST	Master_TIMER compare 1 event resets counter Refer to MTCMP0RST description
5	MTCMP0RST	Master_TIMER compare 0 event resets counter This bit specifies whether the Master_TIMER compare 0 event can reset the counter. 0: Master_TIMER compare 0 event do not reset counter 1: Master_TIMER compare 0 event resets counter
4	MTPERRST	Master_TIMER period event resets counter This bit specifies whether the Master_TIMER period event can reset the counter. 0: Master_TIMER period event do not reset counter 1: Master_TIMER period event resets counter
3	CMP3RST	Slave_TIMER3 compare 3 event resets counter Refer to CMP1RST description
2	CMP1RST	Slave_TIMER3 compare 1 event resets counter This bit specifies whether the compare 1 event can reset the counter. 0: Compare 1 event do not reset counter 1: Compare 1 event resets counter
1	UPRST	Slave_TIMER3 update event resets counter This bit specifies whether the update event can reset the counter. 0: Update event do not reset counter 1: Update event resets counter
0	Reserved	Must be kept at reset value

For Slave_TIMER4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	ST3CMP3RST	ST3CMP1RST	ST3CMP0RST	ST2CMP3RST	ST2CMP1RST	ST2CMP0RST	ST1CMP3RST	ST1CMP1RST	ST1CMP0RST	ST0CMP3RST	ST0CMP1RST	ST0CMP0RST	EXEV9RST	EXEV8RST	EXEV7RST
	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

EXEV6RST	EXEV5RST	EXEV4RST	EXEV3RST	EXEV2RST	EXEV1RST	EXEV0RST	MTCMP3RST	MTCMP2RST	MTCMP1RST	MTCMP0RST	MTPERRST	CMP3RST	CMP1RST	UPRST	Reserved
rw	rw	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	ST3CMP3RST	Slave_TIMER3 compare 3 event resets counter Refer to ST3CMP0RST description.
29	ST3CMP1RST	Slave_TIMER3 compare 1 event resets counter Refer to ST3CMP0RST description.
28	ST3CMP0RST	Slave_TIMER3 compare 0 event resets counter This bit specifies whether the Slave_TIMER3 compare 0 event can reset the counter. 0: Slave_TIMER3 compare 0 event do not reset counter 1: Slave_TIMER3 compare 0 event resets counter
27	ST2CMP3RST	Slave_TIMER2 compare 3 event resets counter Refer to ST2CMP0RST description.
26	ST2CMP1RST	Slave_TIMER2 compare 1 event resets counter Refer to ST2CMP0RST description.
25	ST2CMP0RST	Slave_TIMER2 compare 0 event resets counter This bit specifies whether the Slave_TIMER2 compare 0 event can reset the counter. 0: Slave_TIMER2 compare 0 event do not reset counter 1: Slave_TIMER2 compare 0 event resets counter
24	ST1CMP3RST	Slave_TIMER1 compare 3 event resets counter Refer to ST1CMP0RST description.
23	ST1CMP1RST	Slave_TIMER1 compare 1 event resets counter Refer to ST1CMP0RST description.
22	ST1CMP0RST	Slave_TIMER1 compare 0 event resets counter This bit specifies whether the Slave_TIMER1 compare 0 event can reset the counter. 0: Slave_TIMER1 compare 0 event do not reset counter 1: Slave_TIMER1 compare 0 event resets counter
21	ST0CMP3RST	Slave_TIMER0 compare 3 event resets counter Refer to ST0CMP0RST description.
20	ST0CMP1RST	Slave_TIMER0 compare 1 event resets counter Refer to ST0CMP0RST description.

19	ST0CMP0RST	Slave_TIMER0 compare 0 event resets counter This bit specifies whether the Slave_TIMER0 compare 0 event can reset the counter. 0: Slave_TIMER0 compare 0 event do not reset counter 1: Slave_TIMER0 compare 0 event resets counter
18	EXEV9RST	External event 9 resets counter Refer to EXEV0RST description.
17	EXEV8RST	External event 8 resets counter Refer to EXEV0RST description.
16	EXEV7RST	External event 7 resets counter Refer to EXEV0RST description.
15	EXEV6RST	External event 6 resets counter Refer to EXEV0RST description.
14	EXEV5RST	External event 5 resets counter Refer to EXEV0RST description.
13	EXEV4RST	External event 4 resets counter Refer to EXEV0RST description.
12	EXEV3RST	External event 3 resets counter Refer to EXEV0RST description.
11	EXEV2RST	External event 2 resets counter Refer to EXEV0RST description.
10	EXEV1RST	External event 1 resets counter Refer to EXEV0RST description.
9	EXEV0RST	External event 0 resets counter This bit specifies whether the External event 0 can reset the counter. 0: External event 0 do not reset counter. 1: External event 0 resets counter.
8	MTCMP3RST	Master_TIMER compare 3 event resets counter Refer to MTCMP0RST description
7	MTCMP2RST	Master_TIMER compare 2 event resets counter Refer to MTCMP0RST description
6	MTCMP1RST	Master_TIMER compare 1 event resets counter Refer to MTCMP0RST description
5	MTCMP0RST	Master_TIMER compare 0 event resets counter This bit specifies whether the Master_TIMER compare 0 event can reset the counter. 0: Master_TIMER compare 0 event do not reset counter

		1: Master_TIMER compare 0 event resets counter
4	MTPERRST	Master_TIMER period event resets counter This bit specifies whether the Master_TIMER period event can reset the counter. 0: Master_TIMER period event do not reset counter 1: Master_TIMER period event resets counter
3	CMP3RST	Slave_TIMER4 compare 3 event resets counter Refer to CMP1RST description
2	CMP1RST	Slave_TIMER4 compare 1 event resets counter This bit specifies whether the compare 1 event can reset the counter. 0: Compare 1 event do not reset counter 1: Compare 1 event resets counter
1	UPRST	Slave_TIMER4 update event resets counter This bit specifies whether the update event can reset the counter. 0: Update event do not reset counter 1: Update event resets counter
0	Reserved	Must be kept at reset value

SHRTIMER Slave_TIMERx carrier-signal control register (SHRTIMER_STxCSCTL)

Address offset: 0x58
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					CSFSTPW[3:0]				CSDTY[2:0]			CSPRD[3:0]			
					rw				rw			rw			

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10:7	CSFSTPW[3:0]	First carrier-signal pulse width This bit-field defines the first carrier-signal pulse width following a rising edge on channel output prepare signal(CHxOPRE). $t_{CSFSTPW}$ = (CSFSTPW[3:0]+1) x $t_{SHRTIMER\_CSGCK}$, $t_{SHRTIMER\_CSGCK}$ =16 x $t_{SHRTIMER\_CK}$. 0000: $t_{CSFSTPW}$ = $t_{SHRTIMER\_CSGCK}$ 0001: $t_{CSFSTPW}$ = 2*$t_{SHRTIMER\_CSGCK}$

…

1110: $t_{CSFSTPW}$ = 15*$t_{SHRTIMER\_CSGCK}$

1111: $t_{CSFSTPW}$ = 16*$t_{SHRTIMER\_CSGCK}$

6:4	CSDTY[2:0]	Carrier signal duty cycle This bit-field defines the duty cycle of carrier signal (except the first pulse) which is equal to CSDTY[2:0]/8. 000: 0%(only the first pulse is present). 001: 12.5% 010: 25.0% 011: 37.5% 100: 50.0% 101: 62.5% 110: 75.0% 111: 87.5%
3:0	CSPRD[3:0]	Carrier signal period This bit-field defines the period of carrier signal (except the first pulse). t_{CSPRD} = (CSPRD[3:0]+1) x $t_{SHRTIMER\_CSGCK}$, $t_{SHRTIMER\_CSGCK}$ =16 x $t_{SHRTIMER\_CK}$. 0000: 16 x $t_{SHRTIMER\_CK}$ 0001: 32 x $t_{SHRTIMER\_CK}$ … 1111: 256 x $t_{SHRTIMER\_CK}$

SHRTIMER Slave_TIMERx capture 0 trigger register (SHRTIMER_STxCAP0TRG)

Address offset: 0x5C

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
CP0BST4CMP1	CP0BST4CMP0	CP0BST4NA	CP0BST4A	CP0BST3CMP1	CP0BST3CMP0	CP0BST3NA	CP0BST3A	CP0BST2MP1	CP0BST2CMP0	CP0BST2NA	CP0BST2A	CP0BST1CMP1	CP0BST1CMP0	CP0BST1NA	CP0BST1A
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
CP0BST0CMP1	CP0BST0CMP0	CP0BST0NA	CP0BST0A	CP0BEXEV9	CP0BEXEV8	CP0BEXEV7	CP0BEXEV6	CP0BEXEV5	CP0BEXEV4	CP0BEXEV3	CP0BEXEV2	CP0BEXEV1	CP0BEXEV0	CP0BUP	CP0BSW
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	CP0BST4CMP1	Capture 0 triggered by compare 1 event of Slave_TIMER4 This bit reserved only in Slave_TIMER4. Refer to CP0BST0CMP1 description.
30	CP0BST4CMP0	Capture 0 triggered by compare 0 event of Slave_TIMER4

		This bit reserved only in Slave_TIMER4. Refer to CP0BST0CMP0 description.
29	CP0BST4NA	Capture 0 triggered by ST4CH0_O output active to inactive transition This bit reserved only in Slave_TIMER4. Refer to CP0BST0NA description.
28	CP0BST4A	Capture 0 triggered by ST4CH0_O output inactive to active transition This bit reserved only in Slave_TIMER4. Refer to CP0BST0A description.
27	CP0BST3CMP1	Capture 0 triggered by compare 1 event of Slave_TIMER3 This bit reserved only in Slave_TIMER3. Refer to CP0BST0CMP1 description.
26	CP0BST3CMP0	Capture 0 triggered by compare 0 event of Slave_TIMER3 This bit reserved only in Slave_TIMER3. Refer to CP0BST0CMP0 description.
25	CP0BST3NA	Capture 0 triggered by ST3CH0_O output active to inactive transition This bit reserved only in Slave_TIMER3. Refer to CP0BST0NA description.
24	CP0BST3A	Capture 0 triggered by ST3CH0_O output inactive to active transition This bit reserved only in Slave_TIMER3. Refer to CP0BST0A description.
23	CP0BST2CMP1	Capture 0 triggered by compare 1 event of Slave_TIMER2 This bit reserved only in Slave_TIMER2. Refer to CP0BST0CMP1 description.
22	CP0BST2CMP0	Capture 0 triggered by compare 0 event of Slave_TIMER2 This bit reserved only in Slave_TIMER2. Refer to CP0BST0CMP0 description.
21	CP0BST2NA	Capture 0 triggered by ST2CH0_O output active to inactive transition This bit reserved only in Slave_TIMER2. Refer to CP0BST0NA description.
20	CP0BST2A	Capture 0 triggered by ST2CH0_O output inactive to active transition This bit reserved only in Slave_TIMER2. Refer to CP0BST0A description.
19	CP0BST1CMP1	Capture 0 triggered by compare 1 event of Slave_TIMER1 This bit reserved only in Slave_TIMER1. Refer to CP0BST0CMP1 description.
18	CP0BST1CMP0	Capture 0 triggered by compare 0 event of Slave_TIMER1 This bit reserved only in Slave_TIMER1.

		Refer to CP0BST0CMP0 description.
17	CP0BST1NA	Capture 0 triggered by ST1CH0_O output active to inactive transition This bit reserved only in Slave_TIMER1. Refer to CP0BST0NA description.
16	CP0BST1A	Capture 0 triggered by ST1CH0_O output inactive to active transition This bit reserved only in Slave_TIMER1. Refer to CP0BST0A description.
15	CP0BST0CMP1	Capture 0 triggered by compare 1 event of Slave_TIMER0 This bit reserved only in Slave_TIMER0. 0: Capture 0 is not triggered by compare 1 event of Slave_TIMER0. 1: Capture 0 is triggered by compare 1 event of Slave_TIMER0.
14	CP0BST0CMP0	Capture 0 triggered by compare 0 event of Slave_TIMER0 This bit reserved only in Slave_TIMER0. 0: Capture 0 is not triggered by compare 0 event of Slave_TIMER0. 1: Capture 0 is triggered by compare 0 event of Slave_TIMER0.
13	CP0BST0NA	Capture 0 triggered by ST0CH0_O output active to inactive transition When the Slave_TIMER0 channel 0 output is converted from the active level to the inactive level, capture 0 is triggered by ST0CH0_O. This bit reserved only in Slave_TIMER0. 0: Capture 0 is not triggered by ST0CH0_O output active to inactive transition. 1: Capture 0 is triggered by ST0CH0_O output active to inactive transition.
12	CP0BST0A	Capture 0 triggered by ST0CH0_O output inactive to active transition When the Slave_TIMER0 channel 0 output is converted from the inactive level to the active, capture 0 is triggered by ST0CH0_O. This bit reserved only in Slave_TIMER0. 0: Capture 0 is not triggered by ST0CH0_O output inactive to active transition. 1: Capture 0 is triggered by ST0CH0_O output inactive to active transition.
11	CP0BEXEV9	Capture 0 triggered by external event 9 Refer to CP0BEXEV0 description.
10	CP0BEXEV8	Capture 0 triggered by external event 8 Refer to CP0BEXEV0 description.
9	CP0BEXEV7	Capture 0 triggered by external event 7 Refer to CP0BEXEV0 description.
8	CP0BEXEV6	Capture 0 triggered by external event 6 Refer to CP0BEXEV0 description.
7	CP0BEXEV5	Capture 0 triggered by external event 5 Refer to CP0BEXEV0 description.
6	CP0BEXEV4	Capture 0 triggered by external event 4

		Refer to CP0BEXEV0 description.
5	CP0BEXEV3	Capture 0 triggered by external event 3 Refer to CP0BEXEV0 description.
4	CP0BEXEV2	Capture 0 triggered by external event 2 Refer to CP0BEXEV0 description.
3	CP0BEXEV1	Capture 0 triggered by external event 1 Refer to CP0BEXEV0 description.
2	CP0BEXEV0	Capture 0 triggered by external event 0 When the bit is set, capture 0 is triggered by external event 0 0: Capture 0 is not triggered by external event 0 1: Capture 0 is triggered by external event 0
1	CP0BUP	Capture 0 triggered by update event When the bit is set, capture 0 is triggered by update event 0: Capture 0 is not triggered by update event 1: Capture 0 is triggered by update event
0	CP0BSW	Capture 0 triggered by software This bit can be set by software, and cleared by hardware automatically. When the bit is set, it forces the capture 0 by software. 0: No capture 0 is triggered by software 1: Capture 0 is triggered by software

SHRTIMER Slave_TIMERx capture 1 trigger register (SHRTIMER_STxCAP1TRG)

Address offset: 0x60
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
CP1BST4CMP1	CP1BST4CMP0	CP1BST4NA	CP1BST4A	CP1BST3CMP1	CP1BST3CMP0	CP1BST3NA	CP1BST3A	CP1BST2MP1	CP1BST2CMP0	CP1BST2NA	CP1BST2A	CP1BST1CMP1	CP1BST1CMP0	CP1BST1NA	CP1BST1A
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
CP1BST0CMP1	CP1BST0CMP0	CP1BST0NA	CP1BST0A	CP1BEXEV9	CP1BEXEV8	CP1BEXEV7	CP1BEXEV6	CP1BEXEV5	CP1BEXEV4	CP1BEXEV3	CP1BEXEV2	CP1BEXEV1	CP1BEXEV0	CP1BUP	CP1BSW
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	CP1BST4CMP1	Capture 1 triggered by compare 1 event of Slave_TIMER4 This bit reserved only in Slave_TIMER4.

		Refer to CP1BST0CMP1 description.
30	CP1BST4CMP0	Capture 1 triggered by compare 0 event of Slave_TIMER4 This bit reserved only in Slave_TIMER4. Refer to CP1BST0CMP0 description.
29	CP1BST4NA	Capture 1 triggered by ST4CH0_O output active to inactive transition. This bit reserved only in Slave_TIMER4. Refer to CP1BST0NA description.
28	CP1BST4A	Capture 1 triggered by ST4CH0_O output inactive to active transition This bit reserved only in Slave_TIMER4. Refer to CP1BST0A description.
27	CP1BST3CMP1	Capture 1 triggered by compare 1 event of Slave_TIMER3 This bit reserved only in Slave_TIMER3. Refer to CP1BST0CMP1 description.
26	CP1BST3CMP0	Capture 1 triggered by compare 0 event of Slave_TIMER3 This bit reserved only in Slave_TIMER3. Refer to CP1BST0CMP0 description.
25	CP1BST3NA	Capture 1 triggered by ST3CH0_O output active to inactive transition This bit reserved only in Slave_TIMER3. Refer to CP1BST0NA description.
24	CP1BST3A	Capture 1 triggered by ST3CH0_O output inactive to active transition This bit reserved only in Slave_TIMER3. Refer to CP1BST0A description.
23	CP1BST2CMP1	Capture 1 triggered by compare 1 event of Slave_TIMER2 This bit reserved only in Slave_TIMER2. Refer to CP1BST0CMP1 description.
22	CP1BST2CMP0	Capture 1 triggered by compare 0 event of Slave_TIMER2 This bit reserved only in Slave_TIMER2. Refer to CP1BST0CMP0 description.
21	CP1BST2NA	Capture 1 triggered by ST2CH0_O output active to inactive transition This bit reserved only in Slave_TIMER2. Refer to CP1BST0NA description.
20	CP1BST2A	Capture 1 triggered by ST2CH0_O output inactive to active transition This bit reserved only in Slave_TIMER2. Refer to CP1BST0A description.
19	CP1BST1CMP1	Capture 1 triggered by compare 1 event of Slave_TIMER1 This bit reserved only in Slave_TIMER1. Refer to CP1BST0CMP1 description.

18	CP1BST1CMP0	Capture 1 triggered by compare 0 event of Slave_TIMER1 This bit reserved only in Slave_TIMER1. Refer to CP1BST0CMP0 description.
17	CP1BST1NA	Capture 1 triggered by ST1CH0_O output active to inactive transition This bit reserved only in Slave_TIMER1. Refer to CP1BST0NA description.
16	CP1BST1A	Capture 1 triggered by ST1CH0_O output inactive to active transition This bit reserved only in Slave_TIMER1. Refer to CP1BST0A description.
15	CP1BST0CMP1	Capture 1 triggered by compare 1 event of Slave_TIMER0 This bit reserved only in Slave_TIMER0. 0: Capture 1 is not triggered by compare 1 event of Slave_TIMER0. 1: Capture 1 is triggered by compare 1 event of Slave_TIMER0.
14	CP1BST0CMP0	Capture 1 triggered by compare 0 event of Slave_TIMER0 This bit reserved only in Slave_TIMER0. 0: Capture 1 is not triggered by compare 0 event of Slave_TIMER0. 1: Capture 1 is triggered by compare 0 event of Slave_TIMER0.
13	CP1BST0NA	Capture 1 triggered by ST0CH0_O output active to inactive transition When the Slave_TIMER0 channel 0 output is converted from the active level to the inactive level, capture 1 is triggered by ST0CH0_O. This bit reserved only in Slave_TIMER0. 0: Capture 1 is not triggered by ST0CH0_O output active to inactive transition. 1: Capture 1 is triggered by ST0CH0_O output active to inactive transition.
12	CP1BST0A	Capture 1 triggered by ST0CH0_O output inactive to active transition When the Slave_TIMER0 channel 0 output is converted from the inactive level to the active, capture 1 is triggered by ST0CH0_O. This bit reserved only in Slave_TIMER0. 0: Capture 1 is not triggered by ST0CH0_O output inactive to active transition. 1: Capture 1 is triggered by ST0CH0_O output inactive to active transition.
11	CP1BEXEV9	Capture 1 triggered by external event 9 Refer to CP1BEXEV0 description.
10	CP1BEXEV8	Capture 1 triggered by external event 8 Refer to CP1BEXEV0 description.
9	CP1BEXEV7	Capture 1 triggered by external event 7 Refer to CP1BEXEV0 description.
8	CP1BEXEV6	Capture 1 triggered by external event 6 Refer to CP1BEXEV0 description.
7	CP1BEXEV5	Capture 1 triggered by external event 5

		Refer to CP1BEXEV0 description.
6	CP1BEXEV4	Capture 1 triggered by external event 4 Refer to CP1BEXEV0 description.
5	CP1BEXEV3	Capture 1 triggered by external event 3 Refer to CP1BEXEV0 description.
4	CP1BEXEV2	Capture 1 triggered by external event 2 Refer to CP1BEXEV0 description.
3	CP1BEXEV1	Capture 1 triggered by external event 1 Refer to CP1BEXEV0 description.
2	CP1BEXEV0	Capture 1 triggered by external event 0 When the bit is set, capture 1 is triggered by update external event 0 0: Capture 1 is not triggered by external event 0 1: Capture 1 is triggered by external event 0
1	CP1BUP	Capture 1 triggered by update event When the bit is set, capture 1 is triggered by update event 0: Capture 1 is not triggered by update event 1: Capture 1 is triggered by update event
0	CP1BSW	Capture 1 triggered by software This bit can be set by software, and cleared by hardware automatically. When the bit is set, it forces the capture 1 by software. 0: No capture 1 triggered by software 1: Capture 1 triggered by software

SHRTIMER Slave_TIMERx channel output control register (SHRTIMER_STxCHOCTL)

Address offset: 0x64

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								BMCH1DTI	CH1CSEN	CH1FLTOS[1:0]		ISO1	BMCH1IEN	CH1P	Reserved
								rw	rw	rw		rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DLYISCH[2:0]			DLYISMEN	DTEN	BMCH1DTI	CH0CSEN	CH0FLTOS[1:0]		ISO0	BMCH0IEN	CH0P	Reserved
			rw			rw	rw	rw	rw	rw		rw	rw	rw	

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	BMCH1DTI	Channel 1 dead-time insert in bunch mode In bunch mode, a dead-time can be inserted before output entering the IDLE state. 0: The output enter IDLE immediately. 1: Dead-time is inserted before entering the IDLE state. The deadtime value is set by DTFCFG[15:0]. **Note:** (1) This bit must not be modified once the counter is enabled (STxCEN bit set). (2) This bit can be set only if one of the output idle state is active (ISOy = 1, y=0,1) during IDLE in bunch mode, and the dead-time value is positive (DTFSPROT / DTRSPROT set to 0).
22	CH1CSEN	Channel 1 carrier-signal mode enable 0: Carrier-signal mode of channel 1 is disable. 1: Carrier-signal mode of channel 1 is enable. **Note:** This bit must not be modified once the counter is enabled (STxCEN bit is set).
21:20	CH1FLTOS[1:0]	Channel 1 Fault output state This bit-field specifies the output state when a fault event happened. 00: No effect. The output is normally in Run mode when a fault event happened. 01: The output is active level. 10: The output is inactive level 11: The output is Hi-Z. **Note:** If FLTyEN (y=0..4) in SHRTIMER_STxFLTCTL register is set or the output is in Fault state, this bit-field cannot be modified once the counter is enabled (STxCEN bit is set)
19	ISO1	channel 1 output idle state 0: channel 1 idle state output is inactive. 1: channel 1 idle state output is active. **Note:** This bit must be configured before SHRTIMER control the outputs.
18	BMCH1IEN	Channel 1 IDLE state enable in bunch mode This bit specifies whether channel 1 output can be IDLE state in bunch mode 0: Channel 1 output is not affected by the bunch mode. 1: Channel 1 output can be IDLE state in bunch mode. **Note:** This bit is preloaded and can be changed during run-time, but must not be changed during the bunch mode.
17	CH1P	Channel 1 output polarity This bit specifies the channel 1 output signal polarity. 0: Channel 1 active high 1: Channel 1 active low **Note:** This bit must not be modified once the counter is enabled (STxCEN bit is set)

16:13	Reserved	Must be kept at reset value
12:10	DLYISCH[2:0]	Delayed IDLE source and channel This bit-field specifies the source and channel on which the delayed IDLE state mode are enabled (DLYISMEN = 1). In SHRTIMER_STyCHOCTL(y=0,1,2) register: 000: channel 0 output delayed IDLE on external event 5 001: channel 1 output delayed IDLE on external event 5 010: channel 0 and channel 1 output delayed IDLE on external event 5 011: channel 0 and channel 1 output balanced IDLE on external event 5 in balanced mode (BLNMEN = 1 in SHRTIMER_STyCTL0(y=0,1,2) register) 100: channel 0 output delayed IDLE on external event 6 101: channel 1 output delayed IDLE on external event 6 110: channel 0 and channel 1 delayed IDLE on external event 6 111: channel 0 and channel 1 output balanced IDLE on external event 6 in balanced mode (BLNMEN = 1 in SHRTIMER_STyCTL0(y=0,1,2) register) In SHRTIMER_STyCHOCTL(y=3,4) register: 000: channel 0 output delayed IDLE on external event 7 001: channel 1 output delayed IDLE on external event 7 010: channel 0 and channel 1 output delayed IDLE on external event 7 011: channel 0 and channel 1 output balanced IDLE on external event 7 in balanced mode (BLNMEN = 1 in SHRTIMER_STyCTL0(y=3,4) register) 100: channel 0 output delayed IDLE on external event 8 101: channel 1 output delayed IDLE on external event 8 110: channel 0 and channel 1 output delayed IDLE on external event 8 111: channel 0 and channel 1 output balanced IDLE on external event 8 in balanced mode (BLNMEN = 1 in SHRTIMER_STyCTL0(y=3,4) register) **Note:** This bit-field must not be modified once the delayed IDLE mode is enabled (DLYISMEN bit is set)
9	DLYISMEN	Delayed IDLE state mode enable 0: Delayed IDLE state disable. 1: Delayed IDLE state enable. **Note:** This bit must not be modified once the counter is enabled (STxCEN bit is set)
8	DTEN	Dead time enable 0: Channel 0 and channel 1 outputs are independent. 1: Channel 0 and channel 1 outputs are complementary and dead-time is inserted between channel 0 and channel 1 outputs. **No**te: This bit must not be modified once the counter is enabled (STxCEN bit is set) or its outputs are enabled and controlled by another timer.
7	BMCH0DTI	Channel 0 dead-time insert in bunch mode In bunch mode, a dead-time can be inserted before output entering the IDLE state. 0: The output enter IDLE immediately. 1: Dead-time is inserted before entering the IDLE state. The deadtime value is set

		by DTRCFG[15:0] **Note:** (1) This bit must not be modified once the counter is enabled (STxCEN bit set). (2) This bit can be set only if one of the output idle state is active (ISO0 = 1, y=0,1) during IDLE in bunch mode, and the dead-time value is positive (DTFSPROT / DTRSPROT set to 0).
6	CH0CSEN	Channel 0 carrier-signal mode enable 0: Carrier-signal mode of channel 0 is disable. 1: Carrier-signal mode of channel 0 is enable. **Note:** This bit must not be modified once the counter is enabled (STxCEN bit is set).
5:4	CH0FLTOS[1:0]	Channel 0 Fault output state This bit-field specifies the output state when a fault event happened. 00: No effect. The output is normally in run mode when a fault event happened. 01: The output is active level. 10: The output is inactive level 11: The output is Hi-Z. **Note:** If FLTyEN (y=0..4) in SHRTIMER_STxFLTCTL register is set or the output is in Fault state, this bit-field cannot be modified once the counter is enabled (STxCEN bit is set)
3	ISO0	Channel 0 output idle state 0: Channel 0 idle state output is inactive. 1: Channel 0 idle state output is active. **Note:** This bit must be configured before SHRTIMER control the outputs.
2	BMCH0IEN	Channel 0 IDLE state enable in bunch mode This bit specifies whether channel 0 output can be IDLE state in bunch mode 0: Channel 0 output is not affected by the bunch mode. 1: Channel 0 output can be IDLE state in bunch mode. **Note:** This bit is preloaded and can be changed during run-time, but must not be changed during the bunch mode.
1	CH0P	Channel 0 output polarity This bit specifies the channel 0 output signal polarity. 0: Channel 0 active high 1: Channel 0 active low **Note:** This bit must not be modified once the counter is enabled (STxCEN bit is set)
0	Reserved	Must be kept at reset value

SHRTIMER Slave_TIMERx fault control register (SHRTIMER_STxFLTCTL)

Address offset: 0x68

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
FLTENPROT	Reserved														
rwo															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											FLT4EN	FLT3EN	FLT2EN	FLT1EN	FLT0EN
											rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	FLTENPROT	Protect fault enable This bit-field specifies whether the write protection function is enable or not. This bit is write-once. It can only be cleared by a system reset once It is set by software. 0: Protect disable. FLTyEN (y=0..4) is writable. 1: Protect enable. FLTyEN (y=0..4) is read-only.
30:5	Reserved	Must be kept at reset value
4	FLT4EN	Fault 4 enable 0: Fault 4 disable 1: Fault 4 enable
3	FLT3EN	Fault 3 enable 0: Fault 3 disable 1: Fault 3 enable
2	FLT2EN	Fault 2 enable 0: Fault 2 disable 1: Fault 2 enable
1	FLT1EN	Fault 1 enable 0: Fault 1 disable 1: Fault 1 enable
0	FLT0EN	Fault 0 enable 0: Fault 0 disable 1: Fault 0 enable

SHRTIMER Slave_TIMERx additional control register (SHRTIMER_STxACTL)

Address offset: 0x7C

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

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

Bits	Fields	Descriptions
31:25	DTFCFG[15:9]	Falling edge dead-time value configure This bit-field controls the value of the dead-time following a falling edge of output prepare signal (OyPRE,y=0,1): DTFvalue = DTFCFG[15:0]x $t_{SHRTIMER\_DTGCK}$, $t_{SHRTIMER\_DTGCK}$ = 1/ $f_{SHRTIMER\_DTGCK}$. Writing this register can change the high 7-bit of DTFCFG[15:0]. **Note:** (1) This bit-filed cannot be modified when DTFSVPROT bit in SHRTIMER_STxDTCTL register is set. (2) This bit-filed is preloaded.
24:16	Reserved	Must be kept at reset value
15:9	DTRCFG[15:9]	Rising edge dead-time value configure This bit-field controls the value of the dead-time following a rising edge of output prepare signal (OyPRE,y=0,1): DTRvalue = DTRCFG[15:0]x $t_{SHRTIMER\_DTGCK}$, $t_{SHRTIMER\_DTGCK}$ = 1/ $f_{SHRTIMER\_DTGCK}$. Writing this register can change the high 7-bit of DTRCFG[15:0]. **Note:** (1) This bit-filed cannot be modified when DTRSVPROT bit in SHRTIMER_STxDTCTL register is set. (2) This bit-filed is preloaded.
8:4	Reserved	Must be kept at reset value
3	CNTCKDIV[3]	Counter clock division This bit-field can be configured by software to specify division ratio between the super high resolution clock (SHRTIMER_HPCK) and the counter clock (SHRTIMER_PSCCK). When the CNTCKDIV[3] is '0', $f_{SHRTIMER\_PSCCK}$ = $f_{SHRTIMER\_HPCK}/2^{CNTCKDIV[2:0]+1}$. When the CNTCKDIV[3] bit in the SHRTIMER_MTACTL is '1' and CNTCKDIV[2:0] can only be configured with '3'b000': $f_{SHRTIMER\_PSSCK}$ = $f_{SHRTIMER\_HPCK}$ **Note:** The CNTCKDIV[3:0] bit-field cannot be modified once the timer is enabled
2:0	Reserved	Must be kept at reset value

19.5.3. Common registers

SHRTIMER Master_TIMER registers base address: 0x4001 7780

SHRTIMER control register 0 (SHRTIMER_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				ADTG3USRC[2:0]			ADTG2USRC [2:0]			ADTG1USRC [2:0]			ADTG0USRC [2:0]		
				rw			rw			rw			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										ST4UPDIS	ST3UPDIS	ST2UPDIS	ST1UPDIS	ST0UPDIS	MTUPDIS
										rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:25	ADTG3USRC[2:0]	SHRTIMER_ADCTRIG3 update source This bit-field can be configured by software to specify the source to update the SHRTIMER_ADCTRIGS3 register. 000: Master_TIMER update event 001: Slaver_TIMER0 update event 010: Slaver_TIMER1 update event 011: Slaver_TIMER2 update event 100: Slaver_TIMER3 update event 101: Slaver_TIMER4 update event Other values are reserved
24:22	ADTG2USRC[2:0]	SHRTIMER_ADCTRIG2 update source This bit-field can be configured by software to specify the the source to update the SHRTIMER_ADCTRIGS2 register. 000: Master_TIMER update event 001: Slaver_TIMER0 update event 010: Slaver_TIMER1 update event 011: Slaver_TIMER2 update event 100: Slaver_TIMER3 update event 101: Slaver_TIMER4 update event Other values are reserved
21:19	ADTG1USRC[2:0]	SHRTIMER_ADCTRIG1 update source This bit-field can be configured by software to specify the the source to update the SHRTIMER_ADCTRIGS1 register.

		000: Master_TIMER update event 001: Slaver_TIMER0 update event 010: Slaver_TIMER1 update event 011: Slaver_TIMER2 update event 100: Slaver_TIMER3 update event 101: Slaver_TIMER4 update event Other values are reserved
18:16	ADTG0USRC[2:0]	SHRTIMER_ADCTRIG0 update source This bit-field can be configured by software to specify the the source to update the SHRTIMER_ADCTRIGS0 register. 000: Master_TIMER update event 001: Slaver_TIMER0 update event 010: Slaver_TIMER1 update event 011: Slaver_TIMER2 update event 100: Slaver_TIMER3 update event 101: Slaver_TIMER4 update event Other values are reserved
15:6	Reserved	Must be kept at reset value
5	ST4UPDIS	Slave_TIMER4 update disable This bit is used to enable or disable the update event generation. 0: Update event enable. 1: Update event disable.
4	ST3UPDIS	Slave_TIMER3 update disable This bit is used to enable or disable the update event generation. 0: Update event enable. 1: Update event disable.
3	ST2UPDIS	Slave_TIMER2 update disable This bit is used to enable or disable the update event generation. 0: Update event enable. 1: Update event disable.
2	ST1UPDIS	Slave_TIMER1 update disable This bit is used to enable or disable the update event generation. 0: Update event enable. 1: Update event disable.
1	ST0UPDIS	Slave_TIMER0 update disable This bit is used to enable or disable the update event generation. 0: Update event enable. 1: Update event disable.
0	MTUPDIS	Master_TIMER update disable This bit is used to enable or disable the update event generation.

0: Update event enable.

1: Update event disable.

SHRTIMER control register 1 (SHRTIMER_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		ST4SRST	ST3SRST	ST2SRST	ST1SRST	ST0SRST	MTSRST	Reserved		ST4SUP	ST3SUP	ST2SUP	ST1SUP	ST0SUP	MTSUP
		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	ST4SRST	Slave_TIMER4 software reset This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is reset. 0: No effect. 1: The counter is reset.
12	ST3SRST	Slave_TIMER3 software reset This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is reset. 0: No effect. 1: The counter is reset.
11	ST2SRST	Slave_TIMER2 software reset This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is reset. 0: No effect. 1: The counter is reset.
10	ST1SRST	Slave_TIMER1 software reset This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is reset. 0: No effect. 1: The counter is reset.
9	ST0SRST	Slave_TIMER0 software reset This bit can be set by software, and cleared by hardware automatically. When this

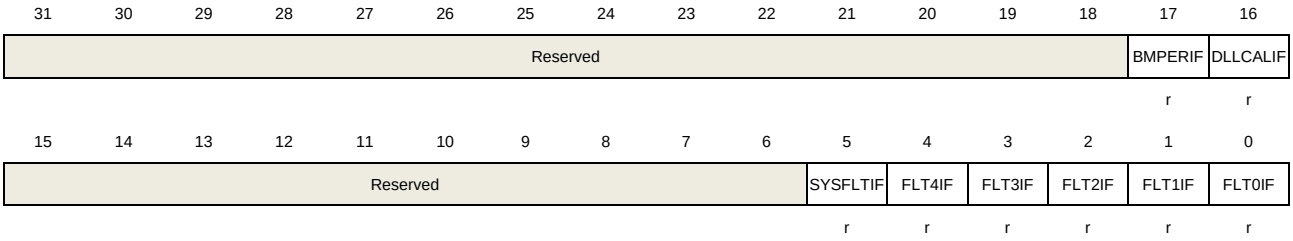
		bit is set, the counter is reset. 0: No effect. 1: The counter is reset.
8	MTSRST	Master_TIMER software reset This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is reset. 0: No effect. 1: The counter is reset.
7:6	Reserved	Must be kept at reset value
5	ST4SUP	Slave_TIMER4 software update This bit can be set by software, and cleared by hardware automatically. When this bit is set, the content of shadow register is transferred to the active register and any pending update request is cancelled. 0: No effect. 1: update generated.
4	ST3SUP	Slave_TIMER3 software update This bit can be set by software, and cleared by hardware automatically. When this bit is set, the content of shadow register is transferred to the active register and any pending update request is cancelled. 0: No effect. 1: update generated.
3	ST2SUP	Slave_TIMER2 software update This bit can be set by software, and cleared by hardware automatically. When this bit is set, the content of shadow register is transferred to the active register and any pending update request is cancelled. 0: No effect. 1: update generated.
2	ST1SUP	Slave_TIMER1 software update This bit can be set by software, and cleared by hardware automatically. When this bit is set, the content of shadow register is transferred to the active register and any pending update request is cancelled. 0: No effect. 1: update generated.
1	ST0SUP	Slave_TIMER0 software update This bit can be set by software, and cleared by hardware automatically. When this bit is set, the content of shadow register is transferred to the active register and any pending update request is cancelled. 0: No effect. 1: update generated.
0	MTSUP	Master_TIMER software update

This bit can be set by software, and cleared by hardware automatically. When this bit is set, the content of shadow register is transferred to the active register and any pending update request is cancelled.
0: No effect.
1: update generated.

SHRTIMER interrupt flag register (SHRTIMER_INTF)

Address offset: 0x08
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value
17	BMPERIF	Bunch mode period interrupt flag This flag is set by hardware when bunch mode period has reached. It is cleared by software writing it at 1. 0: No bunch mode period interrupt occurred 1: Bunch mode period interrupt occurred
16	DLLCALIF	DLL calibration completed interrupt flag This flag is set by hardware when the DLL calibration is completed. It is cleared by software writing it at 1. 0: No DLL calibration completed interrupt occurred 1: DLL calibration completed interrupt occurred
15:6	Reserved	Must be kept at reset value
5	SYSFLTIF	System fault interrupt flag This flag is set by hardware when the system fault occurred. It is cleared by software writing it at 1. 0: No system fault interrupt occurred 1: System fault completed interrupt occurred
4	FLT4IF	Fault 4 interrupt flag Refer to FLT0IF description.

3	FLT3IF	Fault 3 interrupt flag Refer to FLT0IF description.
2	FLT2IF	Fault 2 interrupt flag Refer to FLT0IF description.
1	FLT1IF	Fault 1 interrupt flag Refer to FLT0IF description.
0	FLT0IF	Fault 0 interrupt flag This flag is set by hardware when the fault 0 occurred. It is cleared by software writing it at 1. 0: No fault 0 interrupt occurred 1: Fault 0 completed interrupt occurred

SHRTIMER interrupt flag clear register (SHRTIMER_INTC)

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														BMPERIFC	DLLCALIFC
														w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										SYSFLTIFC	FLT4IFC	FLT3IFC	FLT2IFC	FLT1IFC	FLT0IFC
										w	w	w	w	w	w

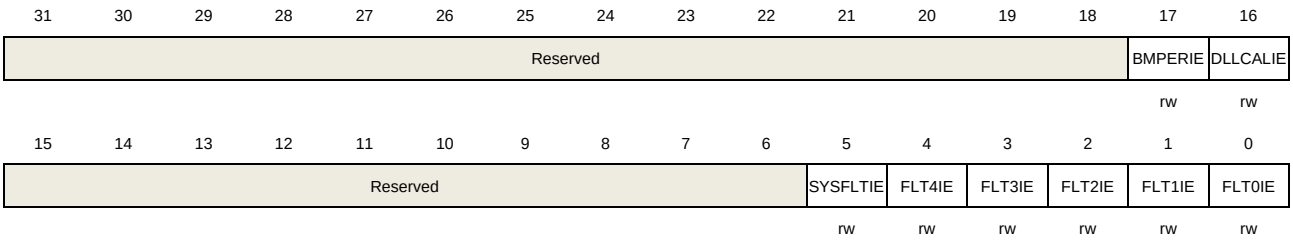
Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value
17	BMPERIFC	Clear bunch mode period interrupt flag Writing 1 to this bit clears the BMPERIF in SHRTIMER_INTF register. 0: No effect 1: Clear bunch mode period interrupt flag
16	DLLCALIFC	Clear DLL calibration completed interrupt flag Writing 1 to this bit clears the DLLCALIF in SHRTIMER_INTF register. 0: No effect 1: Clear DLL calibration completed interrupt flag
15:6	Reserved	Must be kept at reset value
5	SYSFLTIFC	Clear system fault interrupt flag

		Writing 1 to this bit clears the SYSFLTIF in SHRTIMER_INTF register. 0: No effect 1: Clear system fault completed interrupt flag
4	FLT4IFC	Clear fault 4 interrupt flag Writing 1 to this bit clears the FLT4IF in SHRTIMER_INTF register. Refer to FLT0IF description.
3	FLT3IFC	Clear fault 3 interrupt flag Writing 1 to this bit clears the FLT3IF in SHRTIMER_INTF register. Refer to FLT0IF description.
2	FLT2IFC	Clear fault 2 interrupt flag Writing 1 to this bit clears the FLT2IF in SHRTIMER_INTF register. Refer to FLT0IF description.
1	FLT1IFC	Clear fault 1 interrupt flag Writing 1 to this bit clears the FLT1IF in SHRTIMER_INTF register. Refer to FLT0IF description.
0	FLT0IFC	Clear fault 0 interrupt flag Writing 1 to this bit clears the FLT0IF in SHRTIMER_INTF register. 0: No effect 1: Clear fault 0 completed interrupt flag

SHRTIMER interrupt enable register (SHRTIMER_INTEN)

Address offset: 0x10
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value
17	BMPERIE	Bunch mode period interrupt enable 0: Disabled 1: Enabled
16	DLLCALIE	DLL calibration completed interrupt enable

		0: disabled 1: enabled
15:6	Reserved	Must be kept at reset value
5	SYSFLTIE	System fault interrupt enable 0: disabled 1: enabled
4	FLT4IE	Fault 4 interrupt enable Refer to FLT0IE description
3	FLT3IE	Fault 3 interrupt enable Refer to FLT0IE description
2	FLT2IE	Fault 2 interrupt enable Refer to FLT0IE description
1	FLT1IE	Fault 1 interrupt enable Refer to FLT0IE description
0	FLT0IE	Fault 0 interrupt enable 0: disabled 1: enabled

SHRTIMER channel output enable register (SHRTIMER_CHOUTEN)

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						ST4CH1EN	ST4CH0EN	ST3CH1EN	ST3CH0EN	ST2CH1EN	ST2CH0EN	ST1CH1EN	ST1CH0EN	ST0CH1EN	ST0CH0EN
						rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	ST4CH1EN	Slave_TIMER4 channel 1 output (ST4CH1_O) enable Refer to ST0CH0EN description. **Note:** The disable status corresponds to both Idle and Fault states which is given by ST4CH1DISF bit in the SHRTIMER_CHOUTDISF register.

8	ST4CH0EN	Slave_TIMER4 channel 0 output (ST4CH0_O) enable Refer to ST0CH0EN description. **Note:** The disable status corresponds to both Idle and Fault states which is given by ST4CH0DISF bit in the SHRTIMER_CHOUTDISF register.
7	ST3CH1EN	Slave_TIMER3 channel 1 output (ST3CH1_O) enable Refer to ST0CH0EN description. **Note:** The disable status corresponds to both Idle and Fault states which is given by ST3CH1DISF bit in the SHRTIMER_CHOUTDISF register.
6	ST3CH0EN	Slave_TIMER3 channel 0 output (ST3CH0_O) enable Refer to ST0CH0EN description. **Note:** The disable status corresponds to both Idle and Fault states which is given by ST3CH0DISF bit in the SHRTIMER_CHOUTDISF register.
5	ST2CH1EN	Slave_TIMER2 channel 1 output (ST2CH1_O) enable Refer to ST0CH0EN description. **Note:** The disable status corresponds to both Idle and Fault states which is given by ST2CH1DISF bit in the SHRTIMER_CHOUTDISF register.
4	ST2CH0EN	Slave_TIMER2 channel 0 output (ST2CH0_O) enable Refer to ST0CH0EN description. **Note:** The disable status corresponds to both Idle and Fault states which is given by ST2CH0DISF bit in the SHRTIMER_CHOUTDISF register.
3	ST1CH1EN	Slave_TIMER1 channel 1 output (ST1CH1_O) enable Refer to ST0CH0EN description. **Note:** The disable status corresponds to both Idle and Fault states which is given by ST1CH1DISF bit in the SHRTIMER_CHOUTDISF register.
2	ST1CH0EN	Slave_TIMER1 channel 0 output (ST1CH0_O) enable Refer to ST0CH0EN description. **Note:** The disable status corresponds to both Idle and Fault states which is given by ST1CH0DISF bit in the SHRTIMER_CHOUTDISF register.
1	ST0CH1EN	Slave_TIMER0 channel 1 output (ST0CH1_O) enable Refer to ST0CH0EN description. **Note:** The disable status corresponds to both Idle and Fault states which is given by ST0CH1DISF bit in the SHRTIMER_CHOUTDISF register.
0	ST0CH0EN	Slave_TIMER0 channel 0 output (ST0CH0_O) enable Writing 1 to the bit to enable output and writing 0 is has no effect. When fault input is active, the bit is cleared asynchronously by hardware. Reading the bit returns the output enable or disable status. 0: Slave_TIMER0 channel 0 output ST0CH0_O disabled. The output is either in Fault state or Idle state. 1: Slave_TIMER0 channel 0 output ST0CH0_O enabled **Note:** The disable status corresponds to both Idle and Fault states which is given

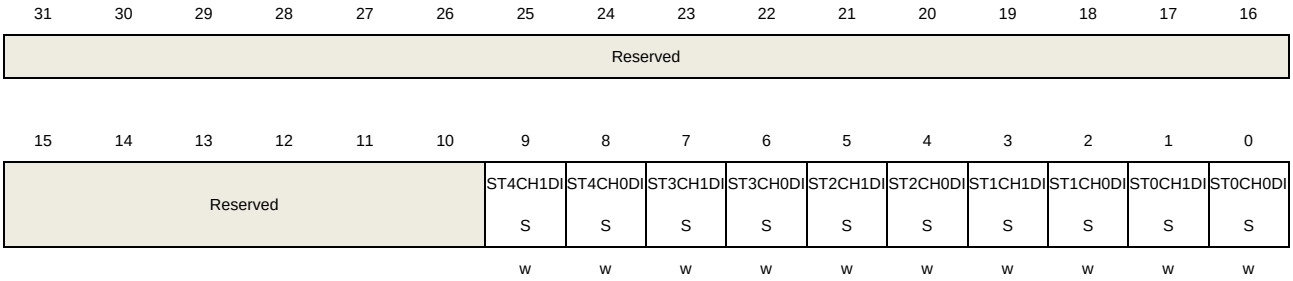
by ST0CH0DISF bit in the SHRTIMER_CHOUTDISF register.

SHRTIMER channel output disable register (SHRTIMER_CHOUTDIS)

Address offset: 0x18

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	ST4CH1DIS	Slave_TIMER4 channel 1 output (ST4CH1_O) disable. Refer to ST0CH0DIS description.
8	ST4CH0DIS	Slave_TIMER4 channel 0 output (ST4CH0_O) disable. Refer to ST0CH0DIS description.
7	ST3CH1DIS	Slave_TIMER3 channel 1 output (ST3CH1_O) disable. Refer to ST0CH0DIS description.
6	ST3CH0DIS	Slave_TIMER3 channel 0 output (ST3CH0_O) disable. Refer to ST0CH0DIS description.
5	ST2CH1DIS	Slave_TIMER2 channel 1 output (ST2CH1_O) disable. Refer to ST0CH0DIS description.
4	ST2CH0DIS	Slave_TIMER2 channel 0 output (ST2CH0_O) disable. Refer to ST0CH0DIS description.
3	ST1CH1DIS	Slave_TIMER1 channel 1 output (ST1CH1_O) disable. Refer to ST0CH0DIS description.
2	ST1CH0DIS	Slave_TIMER1 channel 0 output (ST1CH0_O) disable. Refer to ST0CH0DIS description.
1	ST0CH1DIS	Slave_TIMER0 channel 1 output (ST0CH1_O) disable. Refer to ST0CH0DIS description.
0	ST0CH0DIS	Slave_TIMER0 channel 0 output (ST0CH0_O) disable.

Writing 1 to the bit to disable output and channel 0 enters Idle state. Writing 0 has no effect

0: No effect.

1: Slave_TIMER0 channel 0 output ST0CH0_O disabled. The output entes the Idle state, either from the Fault state or the Run state.

SHRTIMER channel output disable flag register (SHRTIMER_CHOUTDISF)

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						ST4CH1DISF	ST4CH0DISF	ST3CH1DISF	ST3CH0DISF	ST2CH1DISF	ST2CH0DISF	ST1CH1DISF	ST1CH0DISF	ST0CH1DISF	ST0CH0DISF
						r	r	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	ST4CH1DISF	Slave_TIMER4 channel 1 output (ST4CH1_O) disable flag. Refer to ST0CH0DISF description.
8	ST4CH0DISF	Slave_TIMER4 channel 0 output (ST4CH0_O) disable flag. Refer to ST0CH0DISF description.
7	ST3CH1DISF	Slave_TIMER3 channel 1 output (ST3CH1_O) disable flag. Refer to ST0CH0DISF description.
6	ST3CH0DISF	Slave_TIMER3 channel 0 output (ST3CH0_O) disable flag. Refer to ST0CH0DISF description.
5	ST2CH1DISF	Slave_TIMER2 channel 1 output (ST2CH1_O) disable flag. Refer to ST0CH0DISF description.
4	ST2CH0DISF	Slave_TIMER2 channel 0 output (ST2CH0_O) disable flag. Refer to ST0CH0DISF description.
3	ST1CH1DISF	Slave_TIMER1 channel 1 output (ST1CH1_O) disable flag. Refer to ST0CH0DISF description.
2	ST1CH0DISF	Slave_TIMER1 channel 0 output (ST1CH0_O) disable flag. Refer to ST0CH0DISF description.

1	ST0CH1DISF	Slave_TIMER0 channel 1 output (ST0CH1_O) disable flag. Refer to ST0CH0DISF description.
0	ST0CH0DISF	Slave_TIMER0 channel 0 output (ST0CH0_O) disable flag. Reading the bit returns the channel 0 output disable state. It is not significant when the output is enabled. 0: Slave_TIMER0 channel 0 output ST0CH0_O disabled, in Idle state. 1: Slave_TIMER0 channel 0 output ST0CH0_O disabled, in Fault state

SHRTIMER bunch mode control register (SHRTIMER_BMCTL)

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
BMOPTF	Reserved									BMST4	BMST3	BMST2	BMST1	BMST0	BMMT
rc_w0										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					BMSE	BMPSC[3:0]				BMCLKS[3:0]				BMCTN	BMEN
					rw	rw				rw				rw	rw

Bits	Fields	Descriptions
31	BMOPTF	Bunch mode operating flag This flag is set by hardware when the bunch mode is on-going. Writing this bit to 0 will terminate bunch mode. 0: Normal operation. Bunch mode is not operation. 1: Bunch mode operation on-going.
30:22	Reserved	Must be kept at reset value
21	BMST4	Slave_TIMER4 bunch mode 0: Slave_TIMER4 counter clock(SHRTIMER_PSCCK) is maintained and the counter operates normally 1: Slave_TIMER4 counter clock(SHRTIMER_PSCCK) is stopped and the counter is reset **Note:** (1) This bit cannot be changed while the bunch mode is enabled. (2) This bit must not be set when the balanced IDLE mode is active (DLYISCH[2:0] = 3'bx11 in SHRTIMER_STxCHOCTL register).
20	BMST3	Slave_TIMER3 bunch mode 0: Slave_TIMER3 counter clock(SHRTIMER_PSCCK) is maintained and the counter operates normally

		1: Slave_TIMER3 counter clock(SHRTIMER_PSCCK) is stopped and the counter is reset **Note:** (1) This bit cannot be changed while the bunch mode is enabled. (2) This bit must not be set when the balanced IDLE mode is active (DLYISCH[2:0] = 3'bx11 in SHRTIMER_STxCHOCTL register).
19	BMST2	Slave_TIMER2 bunch mode 0: Slave_TIMER2 counter clock(SHRTIMER_PSCCK) is maintained and the counter operates normally 1: Slave_TIMER2 counter clock(SHRTIMER_PSCCK) is stopped and the counter is reset **Note:** (1) This bit cannot be changed while the bunch mode is enabled. (2) This bit must not be set when the balanced IDLE mode is active (DLYISCH[2:0] = 3'bx11 in SHRTIMER_STxCHOCTL register).
18	BMST1	Slave_TIMER1 bunch mode 0: Slave_TIMER1 counter clock(SHRTIMER_PSCCK) is maintained and the counter operates normally 1: Slave_TIMER1 counter clock(SHRTIMER_PSCCK) is stopped and the counter is reset **Note:** (1) This bit cannot be changed while the bunch mode is enabled. (2) This bit must not be set when the balanced IDLE mode is active (DLYISCH[2:0] = 3'bx11 in SHRTIMER_STxCHOCTL register).
17	BMST0	Slave_TIMER0 bunch mode 0: Slave_TIMER0 counter clock(SHRTIMER_PSCCK) is maintained and the counter operates normally 1: Slave_TIMER0 counter clock(SHRTIMER_PSCCK) is stopped and the counter is reset **Note:** (1) This bit cannot be changed while the bunch mode is enabled. (2) This bit must not be set when the balanced IDLE mode is active (DLYISCH[2:0] = 3'bx11 in SHRTIMER_STxCHOCTL register).
16	BMMT	Master_TIMER bunch mode 0: Master_TIMER counter clock(SHRTIMER_PSCCK) is maintained and the counter operates normally 1: Master_TIMER counter clock(SHRTIMER_PSCCK) is stopped and the counter is reset
15:11	Reserved	Must be kept at reset value
10	BMSE	Bunch mode shadow enable 0: The shadow registers for SHRTIMER_BMCMPV and SHRTIMER_BMCAR

registers are disabled

1: The shadow registers for SHRTIMER_BMCMPV and SHRTIMER_BMCAR registers are enabled.

Note: This bit cannot be changed while the bunch mode is enabled

9:6 BMPSC[3:0] Bunch mode clock division

This bit-field can be configured by software to specify division ratio between the SHRTIMER clock (SHRTIMER_CK) and bunch mode counter (SHRTIMER_BMCNTCK) when BMCLKS[3:0] = 4'b1010 in SHRTIMER_BMCTL register.

$f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/2^{BMPSC[3:0]}$

0000: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}$

0001: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/2$

0010: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/4$

0011: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/8$

0100: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/16$

0101: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/32$

0110: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/64$

0111: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/128$

1000: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/256$

1001: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/512$

1010: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/1024$

1011: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/2048$

1100: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/4096$

1101: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/8192$

1110: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/16384$

1111: $f_{SHRTIMER\_BMCNTCK} = f_{SHRTIMER\_CK}/32768$

Note: This bit cannot be changed while the bunch mode is enabled

5:2 BMCLKS[3:0] Bunch mode clock source

This bit-field defines the clock source for the bunch mode counter.

0000: Master_TIMER counter reset/roll-over event.

0001: Slave_TIMER0 counter reset/roll-over event.

0010: Slave_TIMER1 counter reset/roll-over event.

0011: Slave_TIMER2 counter reset/roll-over event.

0100: Slave_TIMER3 counter reset/roll-over event.

0101: Slave_TIMER4 counter reset/roll-over event.

0110: Chip internal signal 0 : BMCLK0

0111: Chip internal signal 1: BMCLK1

1000: Chip internal signal 2: BMCLK2

1001: Chip internal signal 3: BMCLK3

1010: Prescaled $f_{SHRTIMER\_CK}$ clock (as per BMPRSC[3:0] setting)

Other values are reserved

Note:

(1) This bit cannot be changed while the bunch mode is enabled

		(2) BMCLKy (y=0..3): refer to ***Table 19-14. Chip internal signal in bunch mode.***
1	BMCTN	Continuous mode in bunch mode 0: Single pulse mode. The BM-counter stops by hardware when it reaches the SHRTIMER_BMCAR value. 1: Continuous mode. The BM-counter rolls over to zero and counts continuously when it reaches the SHRTIMER_BMCAR value
0	BMEN	Bunch mode enable The bunch mode controller is ready to receive the bunch mode start trigger when the bit is set. Writing this bit to 0 will terminate bunch mode. 0: Bunch mode disable. 1: Bunch mode enable.

SHRTIMER bunch mode start trigger register (SHRTIMER_BMSTRG)

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
CISGN	EXEV7	EXEV6	ST3EXEV7	ST0EXEV6	ST4CMP1	ST4CMP0	ST4REP	ST4RST	ST3CMP1	ST3CMP0	ST3REP	ST3RST	ST2CMP1	ST2CMP0	ST2REP
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
ST2RST	ST1CMP1	ST1CMP0	ST1REP	ST1RST	ST0CMP1	ST0CMP0	ST0REP	ST0RST	MTCMP3	MTCMP2	MTCMP1	MTCMP0	MTREP	MTRST	SWTRG
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	CISGN	Chip internal signal triggers bunch mode operation Chip internal signal (TIMER6_TRGO) is starting the bunch mode operation. 0：No effect on bunch mode operation. 1：Chip internal signal is starting bunch mode operation.
30	EXEV7	External event 7 triggers bunch mode operation External event 7 is starting the bunch mode operation. 0：No effect on bunch mode operation. 1：External event 7 is starting bunch mode operation.
29	EXEV6	External event 6 triggers bunch mode operation External event 6 is starting the bunch mode operation. 0：No effect on bunch mode operation. 1：External event 6 is starting bunch mode operation.
28	ST3EXEV7	Slave_TIMER3 period event following external event 7 triggers bunch mode

		operation Slave_TIMER3 period event following external event 7 is starting the bunch mode operation. 0：No effect on bunch mode operation. 1：Slave_TIMER3 period event following external event 7 is starting bunch mode operation.
27	ST0EXEV6	Slave_TIMER0 period event following external event 6 triggers bunch mode operation Slave_TIMER0 period event following external event 6 is starting the bunch mode operation. 0：No effect on bunch mode operation. 1：Slave_TIMER0 period event following external event 6 is starting bunch mode operation.
26	ST4CMP1	Slave_TIMER4 compare 1 event triggers bunch mode operation Refer to MTCMP1 description.
25	ST4CMP0	Slave_TIMER4 compare 0 event triggers bunch mode operation Refer to MTCMP0 description.
24	ST4REP	Slave_TIMER4 repetition event triggers bunch mode operation Refer to MTREP description.
23	ST4RST	Slave_TIMER4 reset event triggers bunch mode operation Refer to MTRST description.
22	ST3CMP1	Slave_TIMER3 compare 1 event triggers bunch mode operation Refer to MTCMP1 description.
21	ST3CMP0	Slave_TIMER3 compare 0 event triggers bunch mode operation Refer to MTCMP0 description.
20	ST3REP	Slave_TIMER3 repetition event triggers bunch mode operation Refer to MTREP description.
19	ST3RST	Slave_TIMER3 reset event triggers bunch mode operation Refer to MTRST description.
18	ST2CMP1	Slave_TIMER2 compare 1 event triggers bunch mode operation Refer to MTCMP1 description.
17	ST2CMP0	Slave_TIMER2 compare 0 event triggers bunch mode operation Refer to MTCMP0 description.
16	ST2REP	Slave_TIMER2 repetition event triggers bunch mode operation Refer to MTREP description.
15	ST2RST	Slave_TIMER2 reset event triggers bunch mode operation Refer to MTRST description.

14	ST1CMP1	Slave_TIMER1 compare 1 event triggers bunch mode operation Refer to MTCMP1 description.
13	ST1CMP0	Slave_TIMER1 compare 0 event triggers bunch mode operation Refer to MTCMP0 description.
12	ST1REP	Slave_TIMER1 repetition event triggers bunch mode operation Refer to MTREP description.
11	ST1RST	Slave_TIMER1 reset event triggers bunch mode operation Refer to MTRST description.
10	ST0CMP1	Slave_TIMER0 compare 1 event triggers bunch mode operation Refer to MTCMP1 description.
9	ST0CMP0	Slave_TIMER0 compare 0 event triggers bunch mode operation Refer to MTCMP0 description.
8	ST0REP	Slave_TIMER0 repetition event triggers bunch mode operation Refer to MTREP description.
7	ST0RST	Slave_TIMER0 reset event triggers bunch mode operation Refer to MTRST description.
6	MTCMP3	Master_TIMER compare 3 event triggers bunch mode operation Refer to MTCMP0 description.
5	MTCMP2	Master_TIMER compare 2 event triggers bunch mode operation Refer to MTCMP0 description.
4	MTCMP1	Master_TIMER compare 1 event triggers bunch mode operation Refer to MTCMP0 description.
3	MTCMP0	Master_TIMER compare 0 event triggers bunch mode operation The Master_TIMER compare 0 event is starting the bunch mode operation. 0：No effect on bunch mode operation. 1：Master_TIMER compare 0 event is starting bunch mode operation.
2	MTREP	Master_TIMER repetition event triggers bunch mode operation The Master_TIMER repetition event is starting the bunch mode operation. 0：No effect on bunch mode operation. 1：Master_TIMER repetition event is starting bunch mode operation.
1	MTRST	Master_TIMER reset event triggers bunch mode operation The Master_TIMER reset and roll-over event is starting the bunch mode operation. 0：No effect on bunch mode operation. 1：Master_TIMER reset and roll-over event is starting bunch mode operation.
0	SWTRG	Software triggers bunch mode operation This bit is set by software and cleared by hardware automatically. When this bit is set, it will trigger bunch mode operation. This bit is not significant if the bunch mode

is not enabled (BMEN bit in SHRTIMER_BMCTL register is reset).

0：No effect on bunch mode operation.

1：Software trigger is starting bunch mode operation.

Note: This bit is not active if the bunch mode is not enabled (BMEN bit is reset).

SHRTIMER bunch mode compare value register (SHRTIMER_BMCMPV)

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
BMCMPVAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	BMCMPVAL[15:0]	Bunch mode compare value This bit-field contains value to be compared to the BM-counter and defines the duration of the IDLE. This register has a shadow register and can be preloaded. **Note:** BMCMPVAL [15:0] cannot be set to 0x0000 when using the $f_{SHRTIMER\_CK}$ clock with a prescaler equaled to 0 as the bunch mode clock source (BMCLKS[3:0] = 4'b1010 and BMPSC[3:0] = 4'b0000).

SHRTIMER bunch mode counter auto reload register (SHRTIMER_BMCAR)

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
BMCARL[15:0]															
rw															

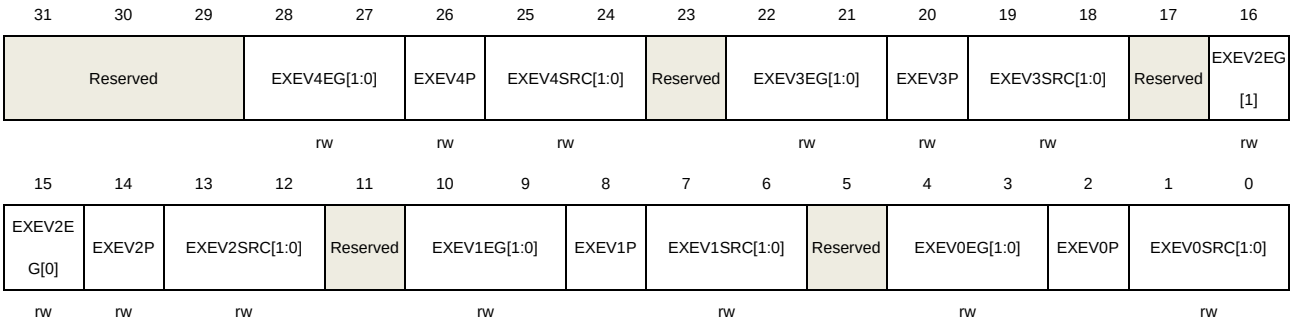
Bits	Fields	Descriptions

31:16	Reserved	Must be kept at reset value
15:0	BMCARL[15:0]	Bunch mode counter auto reload value This bit-field specifies the auto reload value of the BM-counter and defines the bunch mode period which is the sum of the IDLE and RUN duration. This register has a shadow register and can be preloaded. **Note:** This bit-field must not be zero when the burst mode is enabled.

SHRTIMER external event configuration register 0 (SHRTIMER_EXEVCFG0)

Address offset: 0x30
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28:27	EXEV4EG[1:0]	External event 4 edge sensitivity Refer to EXEV0EG[1:0] description.
26	EXEV4P	External event 4 polarity Refer to EXEV0P description.
25:24	EXEV4SRC[1:0]	External event 4 source Refer to EXEV0SRC[1:0] description.
23	Reserved	Must be kept at reset value
22:21	EXEV3EG[1:0]	External event 3 edge sensitivity Refer to EXEV0EG[1:0] description.
20	EXEV3P	External event 3 polarity Refer to EXEV0P description.
19:18	EXEV3SRC[1:0]	External event 3 source Refer to EXEV0SRC[1:0] description.

17	Reserved	Must be kept at reset value
16:15	EXEV2EG[1:0]	External event 2 edge sensitivity Refer to EXEV0EG[1:0] description.
14	EXEV2P	External event 2 polarity Refer to EXEV0P description.
13:12	EXEV2SRC[1:0]	External event 2 source Refer to EXEV0SRC[1:0] description.
11	Reserved	Must be kept at reset value
10:9	EXEV1EG[1:0]	External event 1 edge sensitivity Refer to EXEV0EG[1:0] description.
8	EXEV1P	External event 1 polarity Refer to EXEV0P description.
7:6	EXEV1SRC[1:0]	External event 1 source Refer to EXEV0SRC[1:0] description.
5	Reserved	Must be kept at reset value
4:3	EXEV0EG[1:0]	External event 0 edge sensitivity This bit-field specifies the polarity of external event 0. 00: Level active. Active level is defined by EXEV0P bit 01: Rising edge active, EXEV0P bit is invalid. 10: Falling edge active, EXEV0P bit is invalid 11: Both edges active, EXEV0P bit is invalid.
2	EXEV0P	External event 0 polarity This bit specifies the active level of external event 0 when EXEV0EG[1:0] = 2'b00. 0: External event 0 active at high level. 1: External event 0 active at low level. **Note:** This bit cannot be changed once the Slave_TIMERx is enabled.
1:0	EXEV0SRC[1:0]	External event 0 source 00: External event 0 source is EXEV0SRC 0 01: External event 0 source is EXEV0SRC 1 10: External event 0 source is EXEV0SRC 2 11: External event 0 source is EXEV0SRC 3 **Note:** This bit cannot be changed once the Slave_TIMERx is enabled.

SHRTIMER external event configuration register 1 (SHRTIMER_EXEVCFG1)

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			EXEV9EG[1:0]		EXEV9P	EXEV9SRC[1:0]		Reserved	EXEV8EG[1:0]		EXEV8P	EXEV8SRC[1:0]		Reserved	EXEV7EG[1]
		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
EXEV7EG[0]	EXEV7P	EXEV7SRC[1:0]		Reserved	EXEV6EG[1:0]		EXEV6P	EXEV6SRC[1:0]		Reserved	EXEV5EG[1:0]		EXEV5P	EXEV5SRC[1:0]	
rw	rw	rw			rw			rw			rw			rw	

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28:27	EXEV9EG[1:0]	External event 9 edge sensitivity Refer to EXEV0EG[1:0] in SHRTIMER_EXEVCFG0 register description.
26	EXEV9P	External event 9 polarity Refer to EXEV0P in SHRTIMER_EXEVCFG0 register description.
25:24	EXEV9SRC[1:0]	External event 9 source Refer to EXEV0SRC[1:0] in SHRTIMER_EXEVCFG0 register description.
23	Reserved	Must be kept at reset value
22:21	EXEV8EG[1:0]	External event 8 edge sensitivity Refer to EXEV0EG[1:0] in SHRTIMER_EXEVCFG0 register description.
20	EXEV8P	External event 8 polarity Refer to EXEV0P in SHRTIMER_EXEVCFG0 register description.
19:18	EXEV8SRC[1:0]	External event 8 source Refer to EXEV0SRC[1:0] in SHRTIMER_EXEVCFG0 register description.
17	Reserved	Must be kept at reset value
16:15	EXEV7EG[1:0]	External event 7 edge sensitivity Refer to EXEV0EG[1:0] in SHRTIMER_EXEVCFG0 register description.
14	EXEV7P	External event 7 polarity Refer to EXEV0P in SHRTIMER_EXEVCFG0 register description.
13:12	EXEV7SRC[1:0]	External event 7 source Refer to EXEV0SRC[1:0] in SHRTIMER_EXEVCFG0 register description.
11	Reserved	Must be kept at reset value
10:9	EXEV6EG[1:0]	External event 6 edge sensitivity Refer to EXEV0EG[1:0] in SHRTIMER_EXEVCFG0 register description.
8	EXEV6P	External event 6 polarity

		Refer to EXEV0P in SHRTIMER_EXEVCFG0 register description.
7:6	EXEV6SRC[1:0]	External event 6 source Refer to EXEV0SRC[1:0] in SHRTIMER_EXEVCFG0 register description.
5	Reserved	Must be kept at reset value
4:3	EXEV5EG[1:0]	External event 5 edge sensitivity Refer to EXEV0EG[1:0] in SHRTIMER_EXEVCFG0 register description.
2	EXEV5P	External event 5 polarity Refer to EXEV0P in SHRTIMER_EXEVCFG0 register description.
1:0	EXEV5SRC[1:0]	External event 5 source Refer to EXEV0SRC[1:0] in SHRTIMER_EXEVCFG0 register description.

SHRTIMER external event digital filter control register (SHRTIMER_EXEVDFCTL)

Address offset: 0x38
Reset value: 0x0000 0000

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

Bits	Fields	Descriptions
31:30	EXEVFDIV[2:0]	External event digital filter clock division This bit-field can be configured by software to specify division ratio between the SHRTIMER clock (SHRTIMER_CK) and the external event digital filter clock (SHRTIMER_EXEVFCK). $f_{SHRTIMER\_EXEVFCK} = f_{SHRTIMER\_CK}/2^{EXEVFDIV[2:0]}$ 00: $f_{SHRTIMER\_EXEVFCK} = f_{SHRTIMER\_CK}$ 01: $f_{SHRTIMER\_EXEVFCK} = f_{SHRTIMER\_CK}/2$ 10: $f_{SHRTIMER\_EXEVFCK} = f_{SHRTIMER\_CK}/4$ 11: $f_{SHRTIMER\_EXEVFCK} = f_{SHRTIMER\_CK}/8$
29:28	Reserved	Must be kept at reset value
27:24	EXEV9FC[3:0]	External event 9 filter control Refer to EXEV5FC[3:0] description.

23:22	Reserved	Must be kept at reset value
21:18	EXEV8FC[3:0]	External event 8 filter control Refer to EXEV5FC[3:0] description.
17:16	Reserved	Must be kept at reset value
15:12	EXEV7FC[3:0]	External event 7 filter control Refer to EXEV5FC[3:0] description.
11:10	Reserved	Must be kept at reset value
9:6	EXEV6FC[3:0]	External event 6 filter control Refer to EXEV5FC[3:0] description.
5:4	Reserved	Must be kept at reset value
3:0	EXEV5FC[3:0]	External event 5 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 (f_{SAMP}) used to sample external event and the length of the digital filter applied to external event. 0000: Filter disalble. 0001: $f_{SAMP} = f_{SHRTIMER\_CK}$, N=2. 0010: $f_{SAMP} = f_{SHRTIMER\_CK}$, N=4. 0011: $f_{SAMP} = f_{SHRTIMER\_CK}$, N=8. 0100: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /2, N=6. 0101: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /2, N=8. 0110: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /4, N=6. 0111: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /4, N=8. 1000: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /8, N=6. 1001: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /8, N=8. 1010: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /16, N=5. 1011: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /16, N=6. 1100: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /16, N=8. 1101: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /32, N=5. 1110: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /32, N=6. 1111: $f_{SAMP} = f_{SHRTIMER\_EXEVFCK}$ /32, N=8.

SHRTIMER trigger source 0 to ADC register (SHRTIMER_ADCTRIGS0)

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

TRG0ST4PER	TRG0ST4C3	TRG0ST4C2	TRG0ST4C1	TRG0ST3PER	TRG0ST3C3	TRG0ST3C2	TRG0ST3C1	TRG0ST2PER	TRG0ST2C3	TRG0ST2C2	TRG0ST2C1	TRG0ST1RST	TRG0ST1PER	TRG0ST1C3	TRG0ST1C2
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

TRG0ST1C1	TRG0ST0RST	TRG0ST0PER	TRG0ST0C3	TRG0ST0C2	TRG0ST0C1	TRG0EXEV4	TRG0EXEV3	TRG0EXEV2	TRG0EXEV1	TRG0EXEV0	TRG0MTPER	TRG0MTC3	TRG0MTC2	TRG0MTC1	TRG0MTC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	TRG0ST4PER	SHRTIMER_ADCTRIG0 on Slave_TIMER4 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER4 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER4 period event.
30	TRG0ST4C3	SHRTIMER_ADCTRIG0 on Slave_TIMER4 compare 3 event Refer to TRG0ST4C1 description.
29	TRG0ST4C2	SHRTIMER_ADCTRIG0 on Slave_TIMER4 compare 2 event Refer to TRG0ST4C1 description.
28	TRG0ST4C1	SHRTIMER_ADCTRIG0 on Slave_TIMER4 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER4 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER4 compare 1 event.
27	TRG0ST3PER	SHRTIMER_ADCTRIG0 on Slave_TIMER3 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER3 period event.
26	TRG0ST3C3	SHRTIMER_ADCTRIG0 on Slave_TIMER3 compare 3 event Refer to TRG0ST3C1 description.
25	TRG0ST3C2	SHRTIMER_ADCTRIG0 on Slave_TIMER3 compare 2 event Refer to TRG0ST3C1 description.
24	TRG0ST3C1	SHRTIMER_ADCTRIG0 on Slave_TIMER3 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER3 compare 1 event.

23	TRG0ST2PER	SHRTIMER_ADCTRIG0 on Slave_TIMER2 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER2 period event.
22	TRG0ST2C3	SHRTIMER_ADCTRIG0 on Slave_TIMER2 compare 3 event Refer to TRG0ST2C1 description.
21	TRG0ST2C2	SHRTIMER_ADCTRIG0 on Slave_TIMER2 compare 2 event Refer to TRG0ST2C1 description.
20	TRG0ST2C1	SHRTIMER_ADCTRIG0 on Slave_TIMER2 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER2 compare 1 event.
19	TRG0ST1RST	SHRTIMER_ADCTRIG0 on Slave_TIMER1 reset The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 reset. 1: ADC trigger event generated on SHRTIMER Slave_TIMER1 reset.
18	TRG0ST1PER	SHRTIMER_ADCTRIG0 on Slave_TIMER1 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER1 period event.
17	TRG0ST1C3	SHRTIMER_ADCTRIG0 on Slave_TIMER1 compare 3 event Refer to TRG0ST1C1 description.
16	TRG0ST1C2	SHRTIMER_ADCTRIG0 on Slave_TIMER1 compare 2 event Refer to TRG0ST1C1 description.
15	TRG0ST1C1	SHRTIMER_ADCTRIG0 on Slave_TIMER1 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER1 compare 1 event.
14	TRG0ST0RST	SHRTIMER_ADCTRIG0 on Slave_TIMER0 reset The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 reset.

		1: ADC trigger event generated on SHRTIMER Slave_TIMER0 reset.
13	TRG0ST0PER	SHRTIMER_ADCTRIG0 on Slave_TIMER0 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER0 period event.
12	TRG0ST0C3	SHRTIMER_ADCTRIG0 on Slave_TIMER0 compare 3 event Refer to TRG0ST0C1 description.
11	TRG0ST0C2	SHRTIMER_ADCTRIG0 on Slave_TIMER0 compare 2 event Refer to TRG0ST0C1 description.
10	TRG0ST0C1	SHRTIMER_ADCTRIG0 on Slave_TIMER0 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER0 compare 1 event.
9	TRG0EXEV4	SHRTIMER_ADCTRIG0 on external event 4 Refer to TRG0EXEV0 description.
8	TRG0EXEV3	SHRTIMER_ADCTRIG0 on external event 3 Refer to TRG0EXEV0 description.
7	TRG0EXEV2	SHRTIMER_ADCTRIG0 on external event 2 Refer to TRG0EXEV0 description.
6	TRG0EXEV1	SHRTIMER_ADCTRIG0 on external event 1 Refer to TRG0EXEV0 description.
5	TRG0EXEV0	SHRTIMER_ADCTRIG0 on external event 0 The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER external event 0(EXEV0C). 1: ADC trigger event generated on SHRTIMER external event 0(EXEV0C).
4	TRG0MTPER	SHRTIMER_ADCTRIG0 on Master_TIMER period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Master_TIMER period event. 1: ADC trigger event generated on SHRTIMER Master_TIMER period event.
3	TRG0MTC3	SHRTIMER_ADCTRIG0 on Master_TIMER compare 3 event Refer to TRG0MTC0 description.
2	TRG0MTC2	SHRTIMER_ADCTRIG0 on Master_TIMER compare 2 event

		Refer to TRG0MTC0 description.
1	TRG0MTC1	SHRTIMER_ADCTRIG0 on Master_TIMER compare 1 event Refer to TRG0MTC0 description.
0	TRG0MTC0	SHRTIMER_ADCTRIG0 on Master_TIMER compare 0 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG0. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Master_TIMER compare 0 event. 1: ADC trigger event generated on SHRTIMER Master_TIMER compare 0 event.

SHRTIMER trigger source 1 to ADC register (SHRTIMER_ADCTRIGS1)

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
TRG1ST4RST	TRG1ST4C3	TRG1ST4C2	TRG1ST4C1	TRG1ST3RST	TRG1ST3PER	TRG1ST3C3	TRG1ST3C2	TRG1ST3C1	TRG1ST2RST	TRG1ST2PER	TRG1ST2C3	TRG1ST2C2	TRG1ST2C1	TRG1ST1PER	TRG1ST1C3
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
TRG1ST1C2	TRG1ST1C1	TRG1ST0PER	TRG1ST0C3	TRG1ST0C2	TRG1ST0C1	TRG1EXEV9	TRG1EXEV8	TRG1EXEV7	TRG1EXEV6	TRG1EXEV5	TRG1MTPER	TRG1MTC3	TRG1MTC2	TRG1MTC1	TRG1MTC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	TRG1ST4RST	SHRTIMER_ADCTRIG1 on Slave_TIMER4 reset The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER4 reset . 1: ADC trigger event generated on SHRTIMER Slave_TIMER4 reset.
30	TRG1ST4C3	SHRTIMER_ADCTRIG1 on Slave_TIMER4 compare 3 event Refer to TRG1ST4C1 description.
29	TRG1ST4C2	SHRTIMER_ADCTRIG1 on Slave_TIMER4 compare 2 event Refer to TRG1ST4C1 description.
28	TRG1ST4C1	SHRTIMER_ADCTRIG1 on Slave_TIMER4 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER4 compare 1 event.

		1: ADC trigger event generated on SHRTIMER Slave_TIMER4 compare 1 event.
27	TRG1ST3RST	SHRTIMER_ADCTRIG1 on Slave_TIMER3 reset The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 reset. 1: ADC trigger event generated on SHRTIMER Slave_TIMER3 reset .
26	TRG1ST3PER	SHRTIMER_ADCTRIG1 on Slave_TIMER3 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER3 period event.
25	TRG1ST3C3	SHRTIMER_ADCTRIG1 on Slave_TIMER3 compare 3 event Refer to TRG1ST3C1 description.
24	TRG1ST3C2	SHRTIMER_ADCTRIG1 on Slave_TIMER3 compare 2 event Refer to TRG1ST3C1 description.
23	TRG1ST3C1	SHRTIMER_ADCTRIG1 on Slave_TIMER3 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER3 compare 1 event.
22	TRG1ST2RST	SHRTIMER_ADCTRIG1 on Slave_TIMER2 reset The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 reset . 1: ADC trigger event generated on SHRTIMER Slave_TIMER2 reset.
21	TRG1ST2PER	SHRTIMER_ADCTRIG1 on Slave_TIMER2 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER2 period event.
20	TRG1ST2C3	SHRTIMER_ADCTRIG1 on Slave_TIMER2 compare 3 event Refer to TRG1ST2C1 description.
19	TRG1ST2C2	SHRTIMER_ADCTRIG1 on Slave_TIMER2 compare 2 event Refer to TRG1ST2C1 description.
18	TRG1ST2C1	SHRTIMER_ADCTRIG1 on Slave_TIMER2 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 compare 1

		event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER2 compare 1 event.
17	TRG1ST1PER	SHRTIMER_ADCTRIG1 on Slave_TIMER1 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER1 period event.
16	TRG1ST1C3	SHRTIMER_ADCTRIG1 on Slave_TIMER1 compare 3 event Refer to TRG1ST1C1 description.
15	TRG1ST1C2	SHRTIMER_ADCTRIG1 on Slave_TIMER1 compare 2 event Refer to TRG1ST1C1 description.
14	TRG1ST1C1	SHRTIMER_ADCTRIG1 on Slave_TIMER1 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER1 compare 1 event.
13	TRG1ST0PER	SHRTIMER_ADCTRIG1 on Slave_TIMER0 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER0 period event.
12	TRG1ST0C3	SHRTIMER_ADCTRIG1 on Slave_TIMER0 compare 3 event Refer to TRG1ST0C1 description.
11	TRG1ST0C2	SHRTIMER_ADCTRIG1 on Slave_TIMER0 compare 2 event Refer to TRG1ST0C1 description.
10	TRG1ST0C1	SHRTIMER_ADCTRIG1 on Slave_TIMER0 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER0 compare 1 event.
9	TRG1EXEV9	SHRTIMER_ADCTRIG1 on external event 9 Refer to TRG1EXEV5 description.
8	TRG1EXEV8	SHRTIMER_ADCTRIG1 on external event 8 Refer to TRG1EXEV5 description.
7	TRG1EXEV7	SHRTIMER_ADCTRIG1 on external event 7 Refer to TRG1EXEV5 description.

6	TRG1EXEV6	SHRTIMER_ADCTRIG1 on external event 6 Refer to TRG1EXEV5 description.
5	TRG1EXEV5	SHRTIMER_ADCTRIG1 on external event 5 The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER external event 5 (EXEV5C). 1: ADC trigger event generated on SHRTIMER external event 5 (EXEV5C).
4	TRG1MTPER	SHRTIMER_ADCTRIG1 on Master_TIMER period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Master_TIMER period event. 1: ADC trigger event generated on SHRTIMER Master_TIMER period event.
3	TRG1MTC3	SHRTIMER_ADCTRIG1 on Master_TIMER compare 3 event Refer to TRG1MTC0 description.
2	TRG1MTC2	SHRTIMER_ADCTRIG1 on Master_TIMER compare 2 event Refer to TRG1MTC0 description.
1	TRG1MTC1	SHRTIMER_ADCTRIG1 on Master_TIMER compare 1 event Refer to TRG1MTC0 description.
0	TRG1MTC0	SHRTIMER_ADCTRIG1 on Master_TIMER compare 0 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG1. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Master_TIMER compare 1 event. 1: ADC trigger event generated on SHRTIMER Master_TIMER compare 1 event.

SHRTIMER trigger source 2 to ADC register (SHRTIMER_ADCTRIGS2)

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
TRG2ST4PER	TRG2ST4C3	TRG2ST4C2	TRG2ST4C1	TRG2ST3PER	TRG2ST3C3	TRG2ST3C2	TRG2ST3C1	TRG2ST2PER	TRG2ST2C3	TRG2ST2C2	TRG2ST2C1	TRG2ST1RST	TRG2ST1PER	TRG2ST1C3	TRG2ST1C2
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
TRG2ST1C1	TRG2ST0RST	TRG2ST0PER	TRG2ST0C3	TRG2ST0C2	TRG2ST0C1	TRG2EXEV4	TRG2EXEV3	TRG2EXEV2	TRG2EXEV1	TRG2EXEV0	TRG2MTPER	TRG2MTC3	TRG2MTC2	TRG2MTC1	TRG2MTC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	TRG2ST4PER	SHRTIMER_ADCTRIG2 on Slave_TIMER4 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER4 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER4 period event.
30	TRG2ST4C3	SHRTIMER_ADCTRIG2 on Slave_TIMER4 compare 3 event Refer to TRG2ST4C1 description.
29	TRG2ST4C2	SHRTIMER_ADCTRIG2 on Slave_TIMER4 compare 2 event Refer to TRG2ST4C1 description.
28	TRG2ST4C1	SHRTIMER_ADCTRIG2 on Slave_TIMER4 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER4 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER4 compare 1 event.
27	TRG2ST3PER	SHRTIMER_ADCTRIG2 on Slave_TIMER3 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER3 period event.
26	TRG2ST3C3	SHRTIMER_ADCTRIG2 on Slave_TIMER3 compare 3 event Refer to TRG2ST3C1 description.
25	TRG2ST3C2	SHRTIMER_ADCTRIG2 on Slave_TIMER3 compare 2 event Refer to TRG2ST3C1 description.
24	TRG2ST3C1	SHRTIMER_ADCTRIG2 on Slave_TIMER3 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER3 compare 1 event.
23	TRG2ST2PER	SHRTIMER_ADCTRIG2 on Slave_TIMER2 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER2 period event.
22	TRG2ST2C3	SHRTIMER_ADCTRIG2 on Slave_TIMER2 compare 3 event Refer to TRG2ST2C1 description.

21	TRG2ST2C2	SHRTIMER_ADCTRIG2 on Slave_TIMER2 compare 2 event Refer to TRG2ST2C1 description.
20	TRG2ST2C1	SHRTIMER_ADCTRIG2 on Slave_TIMER2 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER2 compare 1 event.
19	TRG2ST1RST	SHRTIMER_ADCTRIG2 on Slave_TIMER1 reset The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 reset . 1: ADC trigger event generated on SHRTIMER Slave_TIMER1 reset.
18	TRG2ST1PER	SHRTIMER_ADCTRIG2 on Slave_TIMER1 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER1 period event.
17	TRG2ST1C3	SHRTIMER_ADCTRIG2 on Slave_TIMER1 compare 3 event Refer to TRG2ST1C1 description.
16	TRG2ST1C2	SHRTIMER_ADCTRIG2 on Slave_TIMER1 compare 2 event Refer to TRG2ST1C1 description.
15	TRG2ST1C1	SHRTIMER_ADCTRIG2 on Slave_TIMER1 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER1 compare 1 event.
14	TRG2ST0RST	SHRTIMER_ADCTRIG2 on Slave_TIMER0 reset The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 reset . 1: ADC trigger event generated on SHRTIMER Slave_TIMER0 reset.
13	TRG2ST0PER	SHRTIMER_ADCTRIG2 on Slave_TIMER0 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER0 period event.
12	TRG2ST0C3	SHRTIMER_ADCTRIG2 on Slave_TIMER0 compare 3 event

		Refer to TRG2ST0C1 description.
11	TRG2ST0C2	SHRTIMER_ADCTRIG2 on Slave_TIMER0 compare 2 event Refer to TRG2ST0C1 description.
10	TRG2ST0C1	SHRTIMER_ADCTRIG2 on Slave_TIMER0 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER0 compare 1 event.
9	TRG2EXEV4	SHRTIMER_ADCTRIG2 on external event 4 Refer to TRG2EXEV0 description.
8	TRG2EXEV3	SHRTIMER_ADCTRIG2 on external event 3 Refer to TRG2EXEV0 description.
7	TRG2EXEV2	SHRTIMER_ADCTRIG2 on external event 2 Refer to TRG2EXEV0 description.
6	TRG2EXEV1	SHRTIMER_ADCTRIG2 on external event 1 Refer to TRG2EXEV0 description.
5	TRG2EXEV0	SHRTIMER_ADCTRIG2 on external event 0 The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER external event 0 (EXEV0C). 1: ADC trigger event generated on SHRTIMER external event 0 (EXEV0C).
4	TRG2MTPER	SHRTIMER_ADCTRIG2 on Master_TIMER period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Master_TIMER period event. 1: ADC trigger event generated on SHRTIMER Master_TIMER period event.
3	TRG2MTC3	SHRTIMER_ADCTRIG2 on Master_TIMER compare 3 event Refer to TRG2MTC0 description.
2	TRG2MTC2	SHRTIMER_ADCTRIG2 on Master_TIMER compare 2 event Refer to TRG2MTC0 description.
1	TRG2MTC1	SHRTIMER_ADCTRIG2 on Master_TIMER compare 1 event Refer to TRG2MTC0 description.
0	TRG2MTC0	SHRTIMER_ADCTRIG2 on Master_TIMER compare 0 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG2. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Master_TIMER compare 0 event.

1: ADC trigger event generated on SHRTIMER Master_TIMER compare 0 event.

SHRTIMER trigger source 3 to ADC register (SHRTIMER_ADCTRIGS3)

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
TRG3ST4RST	TRG3ST4C3	TRG3ST4C2	TRG3ST4C1	TRG3ST3RST	TRG3ST3PER	TRG3ST3C3	TRG3ST3C2	TRG3ST3C1	TRG3ST2RST	TRG3ST2PER	TRG3ST2C3	TRG3ST2C2	TRG3ST2C1	TRG3ST1PER	TRG3ST1C3
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
TRG3ST1C2	TRG3ST1C1	TRG3ST0PER	TRG3ST0C3	TRG3ST0C2	TRG3ST0C1	TRG3EXEV9	TRG3EXEV8	TRG3EXEV7	TRG3EXEV6	TRG3EXEV5	TRG3MTPER	TRG3MTC3	TRG3MTC2	TRG3MTC1	TRG3MTC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	TRG3ST4RST	SHRTIMER_ADCTRG3 on Slave_TIMER4 reset The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER4 reset . 1: ADC trigger event generated on SHRTIMER Slave_TIMER4 reset.
30	TRG3ST4C3	SHRTIMER_ADCTRG3 on Slave_TIMER4 compare 3 event Refer to TRG3ST4C1 description.
29	TRG3ST4C2	SHRTIMER_ADCTRG3 on Slave_TIMER4 compare 2 event Refer to TRG3ST4C1 description.
28	TRG3ST4C1	SHRTIMER_ADCTRG3 on Slave_TIMER4 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER4 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER4 compare 1 event.
27	TRG3ST3RST	SHRTIMER_ADCTRG3 on Slave_TIMER3 reset The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 reset . 1: ADC trigger event generated on SHRTIMER Slave_TIMER3 reset.
26	TRG3ST3PER	SHRTIMER_ADCTRG3 on Slave_TIMER3 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3.

This bit specifies whether the event can generate the ADC trigger event.
0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 period event.
1: ADC trigger event generated on SHRTIMER Slave_TIMER3 period event.

25 TRG3ST3C3 SHRTIMER_ADCTRG3 on Slave_TIMER3 compare 3 event
Refer to TRG3ST3C1 description.

24 TRG3ST3C2 SHRTIMER_ADCTRG3 on Slave_TIMER3 compare 2 event
Refer to TRG3ST3C1 description.

23 TRG3ST3C1 SHRTIMER_ADCTRG3 on Slave_TIMER3 compare 1 event
The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3.
This bit specifies whether the event can generate the ADC trigger event.
0: No ADC trigger event generated on SHRTIMER Slave_TIMER3 compare 1 event.
1: ADC trigger event generated on SHRTIMER Slave_TIMER3 compare 1 event.

22 TRG3ST2RST SHRTIMER_ADCTRG3 on Slave_TIMER2 reset
The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3.
This bit specifies whether the event can generate the ADC trigger event.
0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 reset .
1: ADC trigger event generated on SHRTIMER Slave_TIMER2 reset.

21 TRG3ST2PER SHRTIMER_ADCTRG3 on Slave_TIMER2 period event
The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3.
This bit specifies whether the event can generate the ADC trigger event.
0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 period event.
1: ADC trigger event generated on SHRTIMER Slave_TIMER2 period event.

20 TRG3ST2C3 SHRTIMER_ADCTRG3 on Slave_TIMER2 compare 3 event
Refer to TRG3ST2C1 description.

19 TRG3ST2C2 SHRTIMER_ADCTRG3 on Slave_TIMER2 compare 2 event
Refer to TRG3ST2C1 description.

18 TRG3ST2C1 SHRTIMER_ADCTRG3 on Slave_TIMER2 compare 1 event
The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3.
This bit specifies whether the event can generate the ADC trigger event.
0: No ADC trigger event generated on SHRTIMER Slave_TIMER2 compare 1 event.
1: ADC trigger event generated on SHRTIMER Slave_TIMER2 compare 1 event.

17 TRG3ST1PER SHRTIMER_ADCTRG3 on Slave_TIMER1 period event
The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3.
This bit specifies whether the event can generate the ADC trigger event.
0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 period event.
1: ADC trigger event generated on SHRTIMER Slave_TIMER1 period event.

16 TRG3ST1C3 SHRTIMER_ADCTRG3 on Slave_TIMER1 compare 3 event

		Refer to TRG3ST1C1 description.
15	TRG3ST1C2	SHRTIMER_ADCTRG3 on Slave_TIMER1 compare 2 event Refer to TRG3ST1C1 description.
14	TRG3ST1C1	SHRTIMER_ADCTRG3 on Slave_TIMER1 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER1 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER1 compare 1 event.
13	TRG3ST0PER	SHRTIMER_ADCTRG3 on Slave_TIMER0 period event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 period event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER0 period event.
12	TRG3ST0C3	SHRTIMER_ADCTRG3 on Slave_TIMER0 compare 3 event Refer to TRG3ST0C1 description.
11	TRG3ST0C2	SHRTIMER_ADCTRG3 on Slave_TIMER0 compare 2 event Refer to TRG3ST0C1 description.
10	TRG3ST0C1	SHRTIMER_ADCTRG3 on Slave_TIMER0 compare 1 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Slave_TIMER0 compare 1 event. 1: ADC trigger event generated on SHRTIMER Slave_TIMER0 compare 1 event.
9	TRG3EXEV9	SHRTIMER_ADCTRG3 on external event 9 Refer to TRG3EXEV5 description.
8	TRG3EXEV8	SHRTIMER_ADCTRG3 on external event 8 Refer to TRG3EXEV5 description.
7	TRG3EXEV7	SHRTIMER_ADCTRG3 on external event 7 Refer to TRG3EXEV5 description.
6	TRG3EXEV6	SHRTIMER_ADCTRG3 on external event 6 Refer to TRG3EXEV5 description.
5	TRG3EXEV5	SHRTIMER_ADCTRG3 on external event 5 The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER external event 5 (EXEV0C). 1: ADC trigger event generated on SHRTIMER external event 5 (EXEV0C).
4	TRG3MTPER	SHRTIMER_ADCTRG3 on Master_TIMER period event

		The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRG3. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Master_TIMER period event. 1: ADC trigger event generated on SHRTIMER Master_TIMER period event.
3	TRG3MTC3	SHRTIMER_ADCTRG3 on Master_TIMER compare 3 event Refer to TRG3MTC0 description.
2	TRG3MTC2	SHRTIMER_ADCTRG3 on Master_TIMER compare 2 event Refer to TRG3MTC0 description.
1	TRG3MTC1	SHRTIMER_ADCTRG3 on Master_TIMER compare 1 event Refer to TRG3MTC0 description.
0	TRG3MTC0	SHRTIMER_ADCTRG3 on Master_TIMER compare 0 event The SHRTIMER can generate an ADC trigger event on SHRTIMER_ADCTRIG3. This bit specifies whether the event can generate the ADC trigger event. 0: No ADC trigger event generated on SHRTIMER Master_TIMER compare 1 event. 1: ADC trigger event generated on SHRTIMER Master_TIMER compare 1 event.

SHRTIMER DLL calibration control register (SHRTIMER_DLLCCTL)

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												CLBPER[1:0]		CLBPEREN	CLBSTRT
												rw		rw	wo

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3:2	CLBPER[1:0]	DLL calibration period This bit-field defines the length of the DLL calibration cycle. 00: 1048576 * $t_{SHRTIMER\_CK}$ 01: 131072 * $t_{SHRTIMER\_CK}$ 10: 16384 * $t_{SHRTIMER\_CK}$ 11: 2048 * $t_{SHRTIMER\_CK}$

1	CLBPEREN	DLL periodic calibration enable This bit enables the periodic DLL calibration. The bit-field CLBPER[1:0] sets the calibration period. 0: DLL periodic calibration disable. 1: DLL periodic calibration enable. **Note:** CLBPEREN bit and CLBSTRT bit must not be set simultaneously.
0	CLBSTRT	DLL calibration start once Writing 1 to the bit starts the DLL calibration when CLBPEREN = 0. This bit is write-only. 0: No effect. 1: DLL calibration start once. **Note:** CLBPEREN bit and CLBSTRT bit must not be set simultaneously.

SHRTIMER fault input configuration register 0 (SHRTIMER_FLTINCFG0)

Address offset: 0x50
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
FLT3INPROT	FLT3INFC[3:0]				FLT3INSRC	FLT3INP	FLT3INEN	FLT2INPROT	FLT2INFC[3:0]				FLT2INSRC	FLT2INP	FLT2INEN
rwo	rw				rw	rw	rw	rwo	rw				rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT1INPROT	FLT1INFC[3:0]				FLT1INSRC	FLT1INP	FLT1INEN	FLT0INPROT	FLT0INFC[3:0]				FLT0INSRC	FLT0INP	FLT0INEN
rwo	rw				rw	rw	rw	rwo	rw				rw	rw	rw

Bits	Fields	Descriptions
31	FLT3INPROT	Protect fault 3 input configuration Refer to FLT0INPROT description.
30:27	FLT3INFC[3:0]	Fault 3 input filter control Refer to FLT0INFC[3:0] description.
26	FLT3INSRC	Fault 3 input source Refer to FLT0INSRC description.
25	FLT3INP	Fault 3 input polarity Refer to FLT0INP description.
24	FLT3INEN	Fault 3 input enable Refer to FLT0INEN description.

23	FLT2INPROT	Protect fault 2 input configuration Refer to FLT0INPROT description.
22:19	FLT2INFC[3:0]	Fault 2 input filter control Refer to FLT0INFC[3:0] description.
18	FLT2INSRC	Fault 2 input source Refer to FLT0INSRC description.
17	FLT2INP	Fault 2 input polarity Refer to FLT0INP description.
16	FLT2INEN	Fault 2 input enable Refer to FLT0INEN description.
15	FLT1INPROT	Protect fault 1 input configuration Refer to FLT0INPROT description.
14:11	FLT1INFC[3:0]	Fault 1 input filter control Refer to FLT0INFC[3:0] description.
10	FLT1INSRC	Fault 1 input source Refer to FLT0INSRC description.
9	FLT1INP	Fault 1 input polarity Refer to FLT0INP description.
8	FLT1INEN	Fault 1 input enable Refer to FLT0INEN description.
7	FLT0INPROT	Protect fault 0 input configuration This bit-field specifies the write protection property to protect fault 0 input configuration. This bit is write-once. It can only be cleared by a system reset once It is set by software. 0: Protect disable. FLT0INEN, FLT0INP, FLT0INSRC and FLT0INFC[3:0] is writable. 1: Protect enable. FLT0INEN, FLT0INP, FLT0INSRC and FLT0INFC[3:0] is read-only.
6:3	FLT0INFC[3:0]	Fault 0 input 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(f_{SAMP}) used to sample external event and the length of the digital filter applied to external event. 0000: Filter disalble. 0001: $f_{SAMP} = f_{SHRTIMER\_CK}$, N=2. 0010: $f_{SAMP} = f_{SHRTIMER\_CK}$, N=4. 0011: $f_{SAMP} = f_{SHRTIMER\_CK}$, N=8. 0100: $f_{SAMP} = f_{SHRTIMER\_FLTFCK}$ /2, N=6. 0101: $f_{SAMP} = f_{SHRTIMER\_FLTFCK}$ /2, N=8.

0110: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /4, N=6.

0111: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /4, N=8.

1000: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /8, N=6.

1001: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /8, N=8.

1010: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /16, N=5.

1011: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /16, N=6.

1100: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /16, N=8.

1101: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /32, N=5.

1110: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /32, N=6.

1111: f_{SAMP} = $f_{SHRTIMER\_FLTFCK}$ /32, N=8.

Note:

(1) This bit-field can be written only when FLT0INEN bit is reset.

(2) This bit-field cannot be modified when FLT0INPROT has been programmed.

2	FLT0INSRC	Fault 0 input source 0: The source of fault 0 input is chip external pin. 1: The source of fault 0 input is chip internal signal(for example comparator) **Note:** This bit can be written only when FLT0INEN bit is reset.
1	FLT0INP	Fault 0 input polarity This bit specifies the polarity of fault 0 input. 0: Fault 0 input active at low level. 1: Fault 0 input active at high level. **Note:** This bit can be written only when FLT0INEN bit is reset.
0	FLT0INEN	Fault 0 input enable This bit can be set to enable fault 0 input circuitry. 0: Fault 0 input disable 1: Fault 0 input enable

SHRTIMER fault input configuration register 1 (SHRTIMER_FLTINCFG1)

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
Reserved						FLTFDIV[2:0]		Reserved							
						rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FLT4INPROT	FLT4INFC[3:0]				FLT4INSRC	FLT4INP	FLT4INEN
								rwo	rw				rw	rw	rw

Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value
25:24	FLTFDIV[2:0]	Fault input digital filter clock division This bit-field can be configured by software to specify division ratio between the SHRTIMER clock (SHRTIMER_CK) and the fault input digital filter clock (SHRTIMER_FLTFCK). $f_{SHRTIMER\_FLTFCK} = f_{SHRTIMER\_CK}/2^{FLTFDIV[2:0]}$. 00: $f_{SHRTIMER\_FLTFCK} = f_{SHRTIMER\_CK}$ 01: $f_{SHRTIMER\_FLTFCK} = f_{SHRTIMER\_CK}/2$ 10: $f_{SHRTIMER\_FLTFCK} = f_{SHRTIMER\_CK}/4$ 11: $f_{SHRTIMER\_FLTFCK} = f_{SHRTIMER\_CK}/8$ **Note:** This bit must be configured before setting any FLTyINEN(y=0..4).
23:8	Reserved	Must be kept at reset value
7	FLT4INPROT	Protect fault 4 input configuration Refer to FLT0INPROT in SHRTIMER_FLTINCFG0 register description.
6:3	FLT4INFC[3:0]	Fault 4 input filter control Refer to FLT0INFC[3:0] in SHRTIMER_FLTINCFG0 register description.
2	FLT4INSRC	Fault 4 input source Refer to FLT0INSRC in SHRTIMER_FLTINCFG0 register description.
1	FLT4INP	Fault 4 input polarity Refer to FLT0INP in SHRTIMER_FLTINCFG0 register description.
0	FLT4INEN	Fault 4 input enable Refer to FLT0INEN in SHRTIMER_FLTINCFG0 register description.

SHRTIMER DMA update Master_TIMER register (SHRTIMER_DMAUPMTR)

Address offset: 0x58
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
MTACTL	Reserved														
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MTCMP3V	MTCMP2V	MTCMP1V	MTCMP0V	MTCREP	MTCAR	MTCNT	MTDMAINTEN	MTINTC	MTCTL0
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	MTACTL	SHRTIMER_MTACTL update by DMA mode Refer to MTCTL0 description.
30:10	Reserved	Must be kept at reset value
9	MTCMP3V	SHRTIMER_MTCMP3V update by DMA mode Refer to MTCTL0 description.
8	MTCMP2V	SHRTIMER_MTCMP2V update by DMA mode Refer to MTCTL0 description.
7	MTCMP1V	SHRTIMER_MTCMP1V update by DMA mode Refer to MTCTL0 description.
6	MTCMP0V	SHRTIMER_MTCMP0V update by DMA mode Refer to MTCTL0 description.
5	MTCREP	SHRTIMER_MTCREP update by DMA mode Refer to MTCTL0 description.
4	MTCAR	SHRTIMER_MTCAR update by DMA mode Refer to MTCTL0 description.
3	MTCNT	SHRTIMER_MTCNT update by DMA mode Refer to MTCTL0 description.
2	MTDMAINTEN	SHRTIMER_MTDMAINTEN update by DMA mode Refer to MTCTL0 description.
1	MTINTC	SHRTIMER_MTINTC update by DMA mode Refer to MTCTL0 description.
0	MTCTL0	SHRTIMER_MTCTL0 update by DMA mode This bit defines if the SHRTIMER_MTCTL0 register is updated by the DMA mode. 0: SHRTIMER_MTCTL0 register is not updated by DMA mode. 1: SHRTIMER_MTCTL0 register is updated DMA mode.

SHRTIMER DMA update Slave_TIMERx register (SHRTIMER_DMAUPSTxR)(x=0..4)

Address offset: 0x5C + x * 0x4, (x=0..4)
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
STxACTL	Reserved										STxFLTCTL	STxCHOCTL	STxCSCTL	STxCNTRST	STxEXEVFCFG1

rw											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STxEXEVFCFG0	STxCH1RST	STxCH1SET	STxCH0RST	STxCH0SET	STxDTCTL	STxCMP3V	STxCMP2V	STxCMP1V	STxCMP0V	STxCREP	STxCAR	STxCNT	STxDMAINTEN	STxINTC	STxCTL0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	STxACTL	SHRTIMER_STxACTL update by DMA mode Refer to STxCTL0 bit description.
30:21	Reserved	Must be kept at reset value
20	STxFLTCTL	SHRTIMER_STxFLTCTL update by DMA mode Refer to STxCTL0 bit description.
19	STxCHOCTL	SHRTIMER_STxCHOCTL update by DMA mode Refer to STxCTL0 bit description.
18	STxCSCTL	SHRTIMER_STxCSCTL update by DMA mode Refer to STxCTL0 bit description.
17	STxCNTRST	SHRTIMER_STxCNTRST update by DMA mode Refer to STxCTL0 bit description.
16	STxEXEVFCFG1	SHRTIMER_STxEXEVFCFG1update by DMA mode Refer to STxCTL0 bit description.
15	STxEXEVFCFG0	SHRTIMER_STxEXEVFCFG0update by DMA mode Refer to STxCTL0 bit description.
14	STxCH1RST	SHRTIMER_STxCH1RST update by DMA mode Refer to STxCTL0 bit description.
13	STxCH1SET	SHRTIMER_STxCH1SET update by DMA mode Refer to STxCTL0 bit description.
12	STxCH0RST	SHRTIMER_STxCH0RST update by DMA mode Refer to STxCTL0 bit description.
11	STxCH0SET	SHRTIMER_STxCH0SET update by DMA mode Refer to STxCTL0 bit description.
10	STxDTCTL	SHRTIMER_STxDTCTL update by DMA mode Refer to STxCTL0 bit description.
9	STxCMP3V	SHRTIMER_STxCMP3V update by DMA mode Refer to STxCTL0 bit description.
8	STxCMP2V	SHRTIMER_STxCMP2V update by DMA mode Refer to STxCTL0 bit description.

7	STxCMP1V	SHRTIMER_STxCMP1V update by DMA mode Refer to STxCTL0 bit description.
6	STxCMP0V	SHRTIMER_STxCMP0V update by DMA mode Refer to STxCTL0 bit description.
5	STxCREP	SHRTIMER_STxCREP update by DMA mode Refer to STxCTL0 bit description.
4	STxCAR	SHRTIMER_STxCAR update by DMA mode Refer to STxCTL0 bit description.
3	STxCNT	SHRTIMER_STxCNT update by DMA mode Refer to STxCTL0 bit description.
2	STxDMAINTEN	SHRTIMER_STxDMAINTEN update by DMA mode Refer to STxCTL0 bit description.
1	STxINTC	SHRTIMER_STxINTC update by DMA mode Refer to STxCTL0 bit description.
0	STxCTL0	SHRTIMER_STxCTL0 update by DMA mode This bit defines if the SHRTIMER_STxCTL0 register is updated by the DMA mode. 0: SHRTIMER_STxCTL0 register is not updated by DMA mode. 1: SHRTIMER_STxCTL0 register is updated DMA mode.

SHRTIMER DMA transfer buffer register (SHRTIMER_DMATB)

Address offset: 0x70
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
DMATB[31:16]															
wo															

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

Bits	Fields	Descriptions
31:0	DMATB[31:0]	DMA transfer buffer When a write operation is assigned to this register, the register selected in SHRTIMER_DMAUPMTR and SHRTIMER_DMAUPSTxR(x=0..4) registers will be accessed. The increment of the register pointer is calculated by hardware.

20. Universal synchronous/asynchronous receiver /transmitter (USART)

20.1. Universal synchronous/asynchronous receiver /transmitter (USARTx, x=0..4)

20.1.1. Overview

The 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 UCLK(PCLK1 or PCLK2) to produce a dedicated baud rate clock 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 polarity of the data bits and the TX/RX pins can be configured independently and flexibly.

All USART supports DMA function for high-speed data communication.

20.1.2. Characteristics

- NRZ standard format.
- Asynchronous, full duplex communication.
- Half duplex single wire communications.
- Programmable baud-rate generator.
  - Divided from the peripheral clocks, PCLK2 for USART0, PCLK1 for USART1/2 and UART3/4.
  - Oversampling by 16 or 8.
  - Maximum speed up to 22.5 MBits/s (PCLK2 180M and oversampling by 8).
- 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).
  - Block mode (T=1).
  - Direct and inverse convention.
- 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), and busy (BSY).
  - 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).
  - Flags for smartcard block mode: end of block (EBF) and receiver timeout (RTF).
  - 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).
- Configurable data polarity.

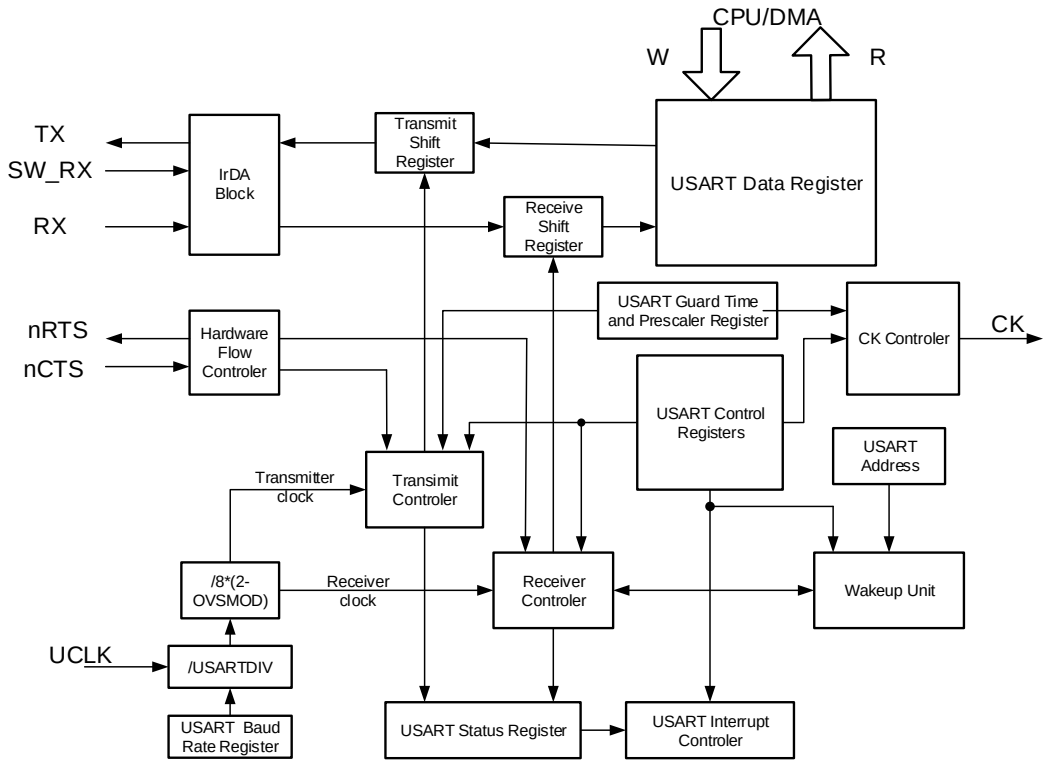
20.1.3. Function overview

The interface is externally connected to another device by the main pins listed in ***Table 20-1. Description of USART important pins***.

Table 20-1. Description of USART important pins

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

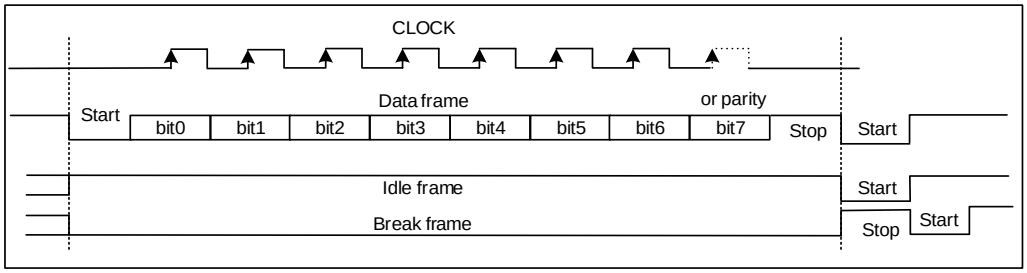
Figure 20-1. USART module block diagram



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 20-2. 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 20-2. Configuration of stop bits

STB[1:0]	stop bit length (bit)	usage description
00	1	Default value

STB[1:0]	stop bit length (bit)	usage description
01	0.5	Smartcard mode for receiving
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.

The break frame structure is a 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 UCLK, the configuration of the baud rate generator and the oversampling mode.

Baud rate generation

The baud-rate divider is a 16-bit number which consists 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 with the peripheral clock:

In case of oversampling by 16, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{16 \times \text{Baud Rate}} \quad (20\text{-}1)$$

In case of oversampling by 8, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{8 \times \text{Baud Rate}} \quad (20\text{-}2)$$

For example, when oversampled by 16:

1. Get USARTDIV by calculating the value of USART_BUAD:
   If USART_BUAD=0x21D, then INTDIV=33 (0x21), FRADIV=13 (0xD).
   USARTDIV=33+13/16=33.81.
2. Get the value of USART_BUAD by calculating the value of USARTDIV:
   If USARTDIV=30.37, then INTDIV=30 (0x1E).
   16*0.37=5.92, the nearest integer is 6, so FRADIV=6 (0x6).
   USART_BUAD=0x1E6.

   Note: If the roundness of FRADIV is 16 (overflow), the carry must be added to the integer part.

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. The polarity of the TX pin can be configured by the TINV bit in the USART_CTL3 register. 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 cleared while the transmission is ongoing.

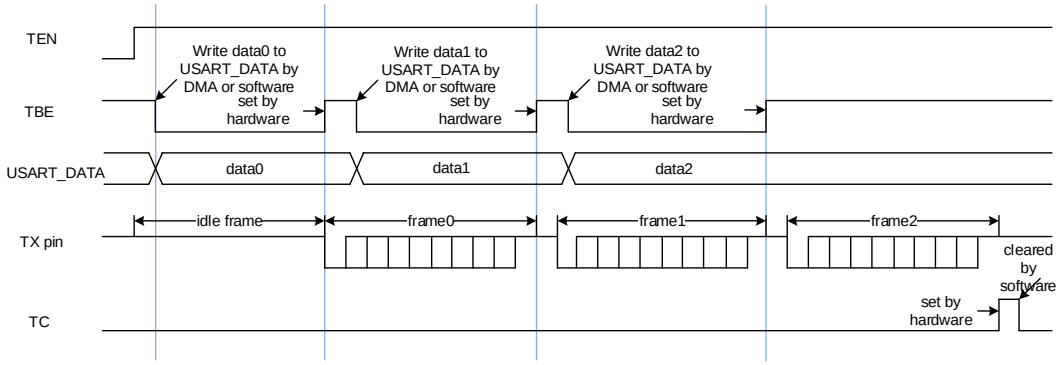
After power on, the TBE bit is high by default. Data can be written to the USART_DATA when the TBE bit in the USART_STAT0 register is asserted. The TBE bit is cleared by writing to the USART_DATA register and it is set by hardware after the data is put into the transmit shift register. If a data is written 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_STAT0 register will be set. An interrupt will be generated if the corresponding interrupt enable bit (TCIE) is set in the USART_CTL0 register.

The USART transmit procedure is shown in ***Figure 20-3. USART transmit procedure***. The software operating process is as follows:

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 to be asserted.
8. Write the data to the USART_DATA register.
9. Repeat step7-8 for each data, if DMA is not enabled.
10. Wait until TC=1 to finish.

Figure 20-3. USART transmit procedure



It is necessary to wait for the TC bit to be asserted before disabling the USART or entering the power saving mode. This bit can be cleared by a software sequence: reading the USART_STAT0 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 directly.

USART receiver

After power on, the USART receiver can be enabled by the following 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 to configure the number of stop bits.
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_STAT0 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 reception are stored in the USART_STAT0 register.

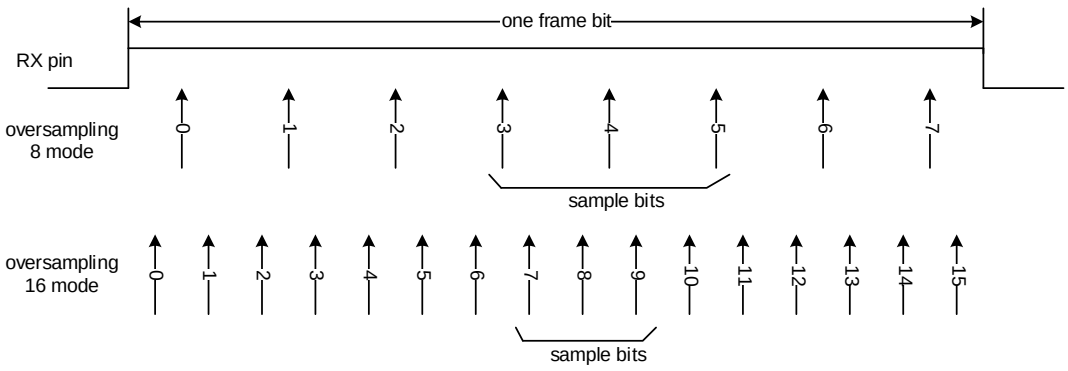
The software can get the received data by reading the USART_DATA register directly, or through DMA. The RBNE bit is cleared by a read operation on the USART_DATA register, whatever it is performed by software directly, or through DMA.

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. If the oversampling 8 mode is enabled, the 3rd, 4th and 5th samples are used, 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 value of the three samples of any bit are not the same, whatever it is a start bit, data bit, parity bit or stop bit, a noisy error (NERR) 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.

The one bit receive mode can be enabled by setting the OBS bit in USART_CTL2. This mode can be enabled when the RX line is noise free and the NERR bit will never be set in this mode.

Figure 20-4. Receiving a frame bit by oversampling method



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_STAT0 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_STAT0 register will be set. An interrupt will be 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_STAT0 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.

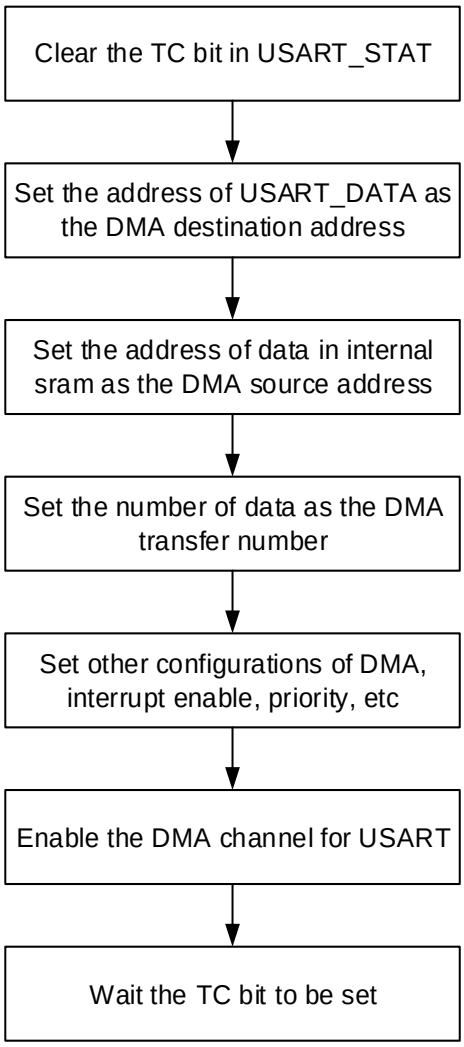
If a noise error (NERR), parity error (PERR), frame error (FERR) or overrun error (ORERR) is generated during a receiving process, then NERR, PERR, FERR or ORERR will be set at same time with RBNE. If DMA is disabled, the software needs to check whether the RBNE interrupt is caused by noise error, parity error, framing error or overflow error when the RBNE interrupt occurs.

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 steps are shown in ***Figure 20-5. Configuration step when using DMA for USART transmission***.

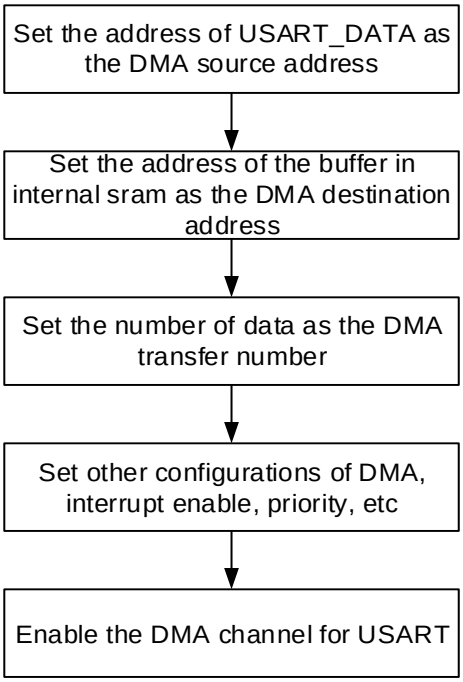
Figure 20-5. Configuration step when using DMA for USART transmission



After all of the data frames are transmitted, the TC bit in USART_STAT0 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 steps are shown in ***Figure 20-6. Configuration steps 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_STAT0.

Figure 20-6. Configuration steps when using DMA for USART reception

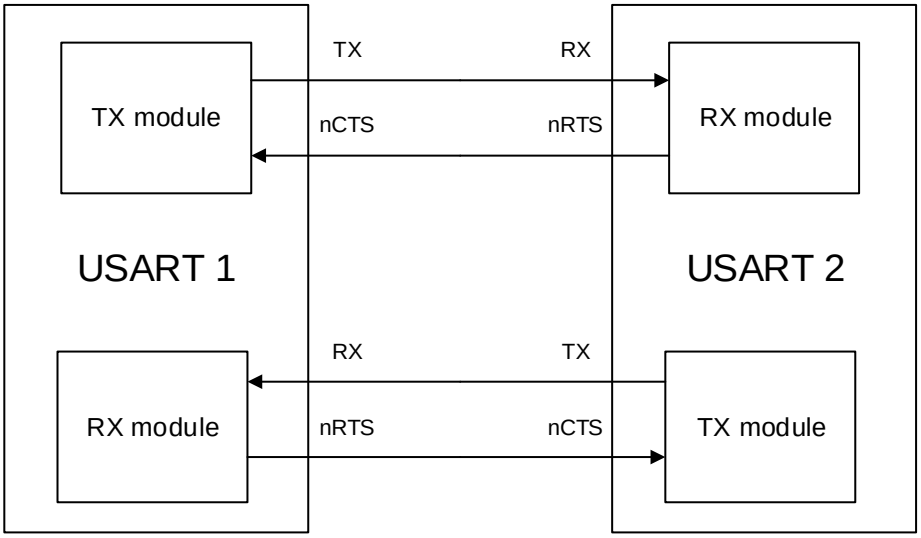


When the number of the data received by USART reaches the DMA transfer number, an end of transfer interrupt will be generated in the DMA module.

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 20-7. Hardware flow control between two USARTs



RTS flow control

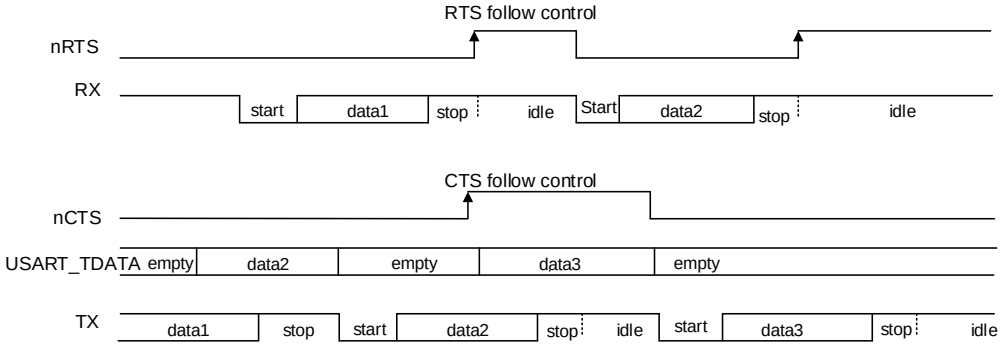
The USART receiver outputs the nRTS, which reflects the status of the receive buffer. When

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 whether a data frame can be transmitted. If the TBE bit in USART_STAT0 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 20-8. Hardware flow control



If the CTS flow control is enabled, the CTSF bit in USART_STAT0 is set when the nCTS pin toggles. An interrupt is generated if the CTSIE bit in USART_CTL2 is set.

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 clearing the RWU bit.

The USART can also be woken 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 it is woken up by an idle frame, the IDLEF bit in USART_STAT0 will 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 will clear the RWU bit and exits the mute mode. The RBNE bit will be set when the frame that wakes up the USART. The status bits are available in the USART_STAT0 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.

LIN mode

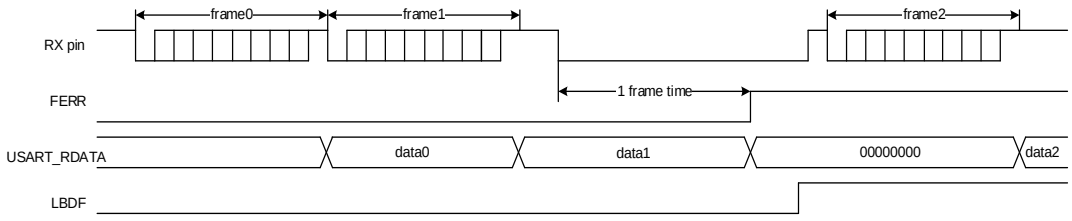
The local interconnection network mode is enabled by setting the LMEN bit in USART_CTL1. The CKEN, WL, STB[1:0] bits in USART_CTL1 and the SCEN, HDEN, IREN bits in USART_CTL2 should be cleared in LIN mode.

When transmitting a normal data frame, the transmission procedure is the same as the normal USART mode. When the SBKCMD bit in USART_CTL0 is set, the USART transmits 13 '0' bits continuously, followed by 1 stop bit.

The break detection function is totally independent of 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 configuring LBLEN bit 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 bit in USART_STAT0 is set. An interrupt occurs if the LBDIE bit in USART_CTL1 is set.

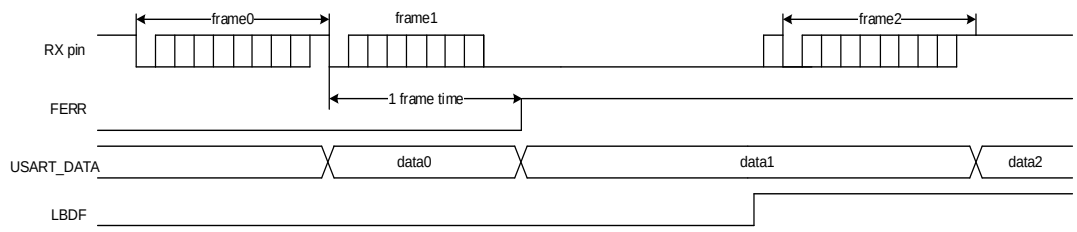
As shown in ***Figure 20-9. Break frame occurs during idle state***, if a break frame occurs during the idle state on the RX pin, the USART receiver will receive an all '0' frame, with an asserted FERR status.

Figure 20-9. Break frame occurs during idle state



As shown in ***Figure 20-10. 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 20-10. Break frame occurs during a frame



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 cleared in synchronous mode. The CK pin is the clock output of the synchronous USART transmitter, and can be only activated when the TEN bit is enabled. No clock pulse will be sent through the CK pin during the transmission of the start bit and stop bit. 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 20-11. Example of USART in synchronous mode

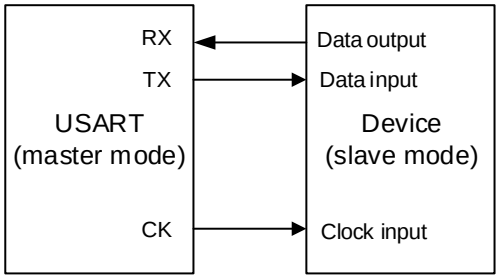
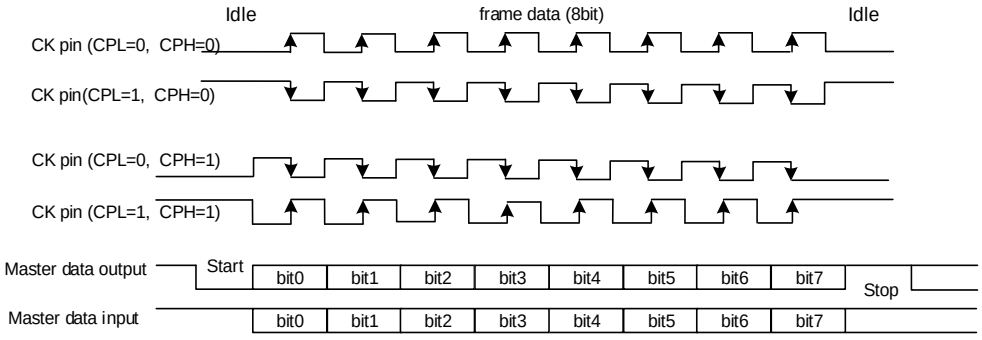


Figure 20-12. 8-bit format USART synchronous waveform (CLEN=1)

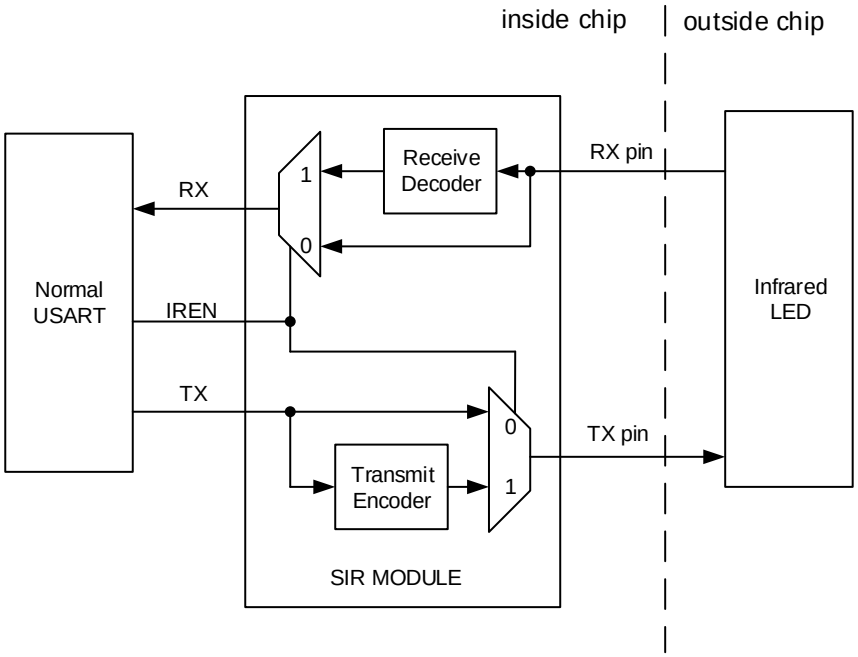


IrDA SIR ENDEC mode

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 cleared 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.

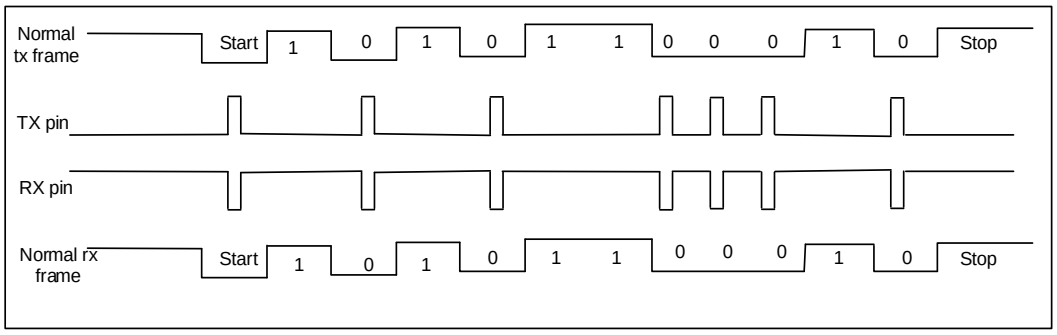
Figure 20-13. IrDA SIR ENDEC module



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 20-14. IrDA data modulation



The SIR sub module 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

division 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 period. The receiver decoder works in the same manner as the normal IrDA mode.

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 cleared 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 that the transmission and reception process never conflict with each other.

The collision detection can be enabled in some applications, such as data transfer over a single data line shared by several sending devices. This function can be enabled by setting the CDEN bit in the USART_GDCTL register. The CD bit will be set when there is a data collision on the txd line and an interrupt will be generated if the CDIE bit is set.

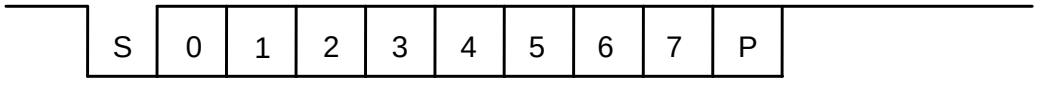
Smartcard (ISO7816-3) mode

The smartcard mode is an asynchronous mode, which is designed to support the ISO7816-3 protocol. Both the character (T=0) mode and the block (T=1) mode are supported. 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 cleared in smartcard mode.

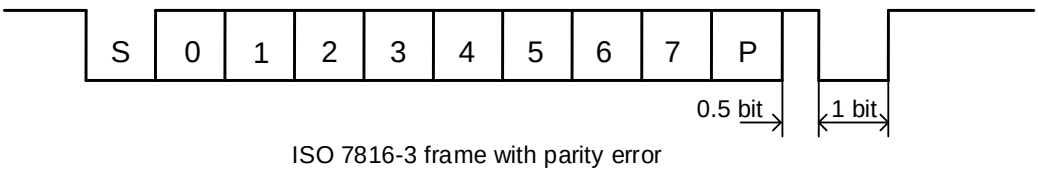
A clock is provided to the external smartcard through the CK pin after the CKEN bit is set. The clock is divided from the PCLK. The division ratio is configured by the PSC[4:0] bits in USART_GP register. The CK pin only provides a clock source to the smartcard.

The smartcard mode is a half-duplex communication protocol. When connected to a smartcard, the TX pin must be configured as open drain mode, 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 20-15. ISO7816-3 frame format



ISO 7816-3 frame without parity error



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 counting 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 for SCRTNUM times. An interframe gap of 2.5 bits time will be inserted before the start of a resented frame. At the end of the last repeated character the TC bit is set immediately without guard time. The USART will stop transmitting and assert the framing error status if it still receives the NACK signal after the programmed number of retries. 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 smartcard. Then a frame error occurs in smartcard side. The RBNE/receive DMA request is not activated if the received character is erroneous. According to the protocol, the smartcard can resend the data. The USART stops transmitting the NACK and the error is regarded as a parity error if the received character is still erroneous after the maximum number of retries which is specified in the SCRTNUM bit field. The NACK signal is enabled by setting the NKEN bit in USART_CTL2.

The idle frame and break frame are not supported in the smartcard mode.

Block (T=1) mode

In block (T=1) mode, the NKEN bit in the USART_CTL2 register should be cleared to deactivate the NACK transmission.

When requesting a read from the smartcard, the RT[23:0] bits in USART_RT register should be programmed with the BWT (block wait time) - 11 value and RBNEIE must be set. This timeout period is expressed in baud time units. The RTF bit in USART_STAT1 will be asserted, if no answer is received from the card before the expiration of this period. An interrupt is generated if the RTIE bit in USART_CTL3 is set. The USART generates a RBNE interrupt if the first character is received before the expiration of the RT[23:0] period. If DMA is used to read from the smartcard in block mode, the DMA must be enabled only after the first character is received.

After the first character is received, the RT[23:0] bits should be configured to the CWT (character wait time) - 11 to enable the automatic check of the maximum interframe gap between two consecutive characters. The RTF bit in USART_STAT1 will be asserted, if the smartcard stops sending characters in the RT[23:0] period.

The USART uses a block length counter, which is reset when the USART is transmitting (TBE=0), to count the number of received characters. The length of the block, which must be programmed to the BL[7:0] bits in the USART_RT register, is received from the smartcard in the third byte of the block (prologue field). The block length counter counts up from 0 to the maximum value of BL[7:0]+4. The end of the block status (EBF bit in USART_STAT1) is set after the block length counter reaches the maximum value. An interrupt is generated if the EBIE bit in USART_CTL3 is set. The RTF bit may be set in case that an error in the block length.

If DMA is used for reception, this register field must be programmed to the minimum value (0x0) before the start of the block. With this value, the end of the block interrupt occurs after the 4th received character. The block length value can be read from the receive buffer at the third byte.

If DMA is not used for reception, the BL[7:0] bits should be firstly configured with the maximum value 0xFF to avoid generating an EBF status. The real block length value can be reconfigured to the BL[7:0] bits after the third byte is received.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

When the directed convention is selected, the LSB of the data frame is transferred first, high state on the TX pin represents logic '1', the parity check mode is even. In this case the MSBF and DINV bits in USART_CTL3 should be cleared.

When the inverse convention is selected, the MSB of the data frame is transferred first, high state on the TX pin represents logic '0', the parity check mode is even. In this case the MSBF and DINV bits in USART_CTL3 should be set.

USART interrupts

The USART interrupt events and flags are listed in ***Table 20-3. USART interrupt requests***.

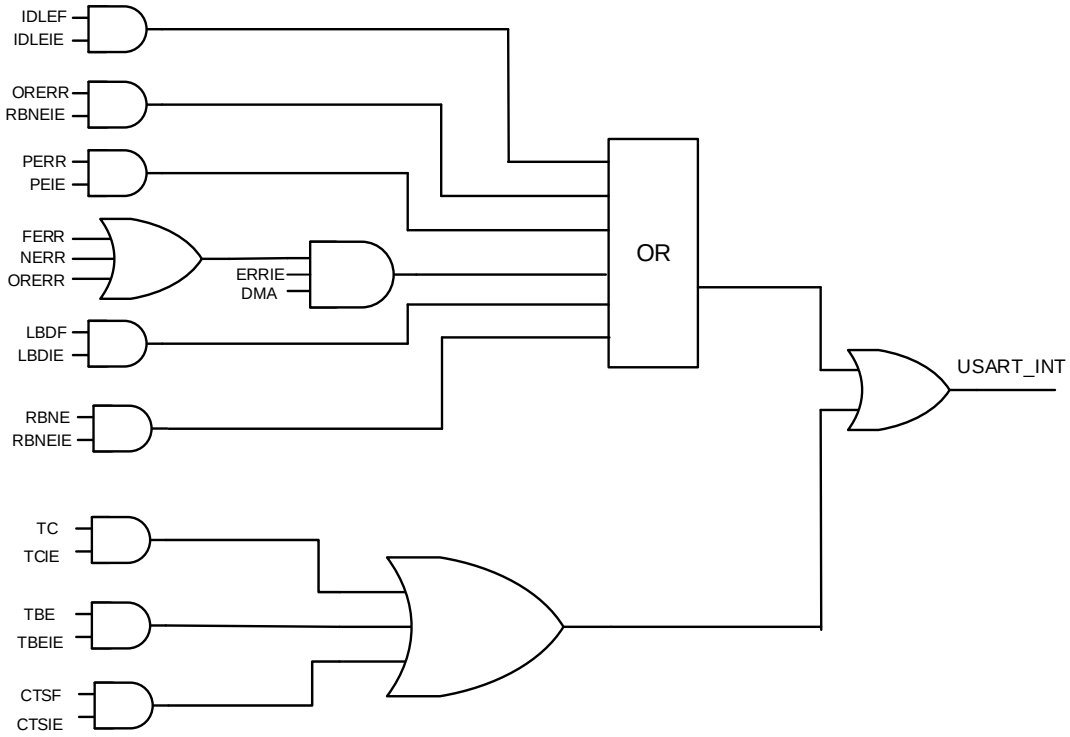
Table 20-3. USART interrupt requests

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
Receiver timeout	RTF	USART_CTL3	RTIE
End of Block	EBF	USART_CTL3	EBIE

Interrupt event	Event flag	Control register	Enable Control bit
Reception errors (noise flag, overrun error, framing error) in DMA reception	NERR or ORERR or FERR	USART_CTL2	ERRIE
Collision detected	CD	USART_GDCTL	CDIE

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 20-16. USART interrupt mapping diagram



20.1.4. Register definition

USART0 base address: 0x4001 3800
USART1 base address: 0x4000 4400
USART2 base address: 0x4000 4800
UART3 base address: 0x4000 4C00
UART4 base address: 0x4000 5000

Status register 0 (USART_STAT0)

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 be kept 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 reserved for UART3/4.
8	LBDF	LIN break detected flag LMEN bit in USART_CTL1 is set 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_STAT0 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_STAT0 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_STAT0 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_STAT0 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_STAT0 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.

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 be kept 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.

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 be kept 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.

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
OVSMOD	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	rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept the reset value.
15	OVSMOD	Oversampe mode 0: oversampling by 16. 1: oversampling by 8. If SCEN=1, IREN=1 or LMEN=1, OVS8 is forced to ‘0’ by hardware.
14	Reserved	Must be kept 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_STAT0 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_STAT0 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_STAT0 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_STAT0 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_STAT0 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. 1: Transmit a break frame.

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 be kept the reset value.
14	LMEN	LIN mode enable 0: LIN mode disabled. 1: LIN mode enabled.
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 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 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 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 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 is reserved for UART3/4.
7	Reserved	Must be kept the reset value.
6	LBDIE	LIN break detected interrupt enable. If this bit is set, an interrupt occurs when the LBDF bit in USART_STAT0 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
4	Reserved	Must be kept 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.

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				OSB	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
31:12	Reserved	Must be kept the reset value.
11	OSB	One sample bit method

		This bit selects the sample method. The noise detection flag (NF) is disabled when the one sample bit method is selected. 0: Three sample bit method. 1: One sample bit method.
10	CTSIE	CTS interrupt enable If this bit is set, an interrupt occurs when the CTSF bit in USART_STAT0 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 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 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 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 is reserved for UART3/4.
3	HDEN	Half-duplex enable This bit enables the half-duplex USART mode. 0: Half duplex mode is disabled.

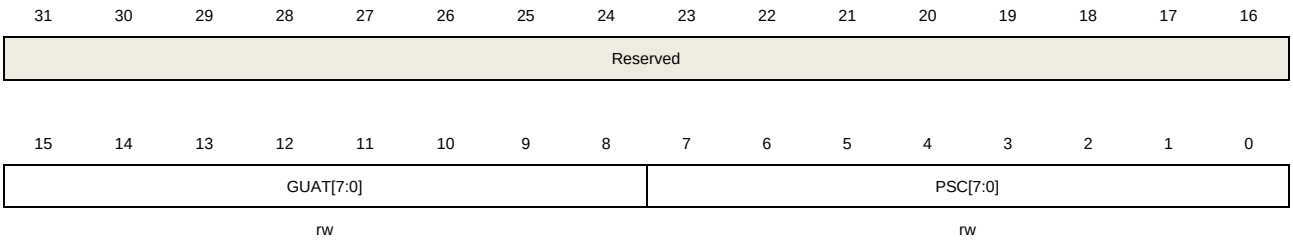
		1: Half duplex mode is enabled.
2	IRLP	IrDA low-power This bit selects low-power mode of IrDA mode. 0: Normal mode. 1: Low-power mode.
1	IREN	IrDA mode enable This bit enables the IrDA mode of USART. 0: IrDA disabled. 1: IrDA enabled.
0	ERRIE	Error interrupt enable When DMA request for reception is enabled (DENR=1), if this bit is set, an interrupt occurs when any one of the FERR, ORERR and NERR bits in USART_STAT0 is set. 0: Error interrupt disabled. 1: Error interrupt enabled.

Guard time and prescaler register (USART_GP)

Address offset: 0x18

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:16	Reserved	Must be kept 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. These bits are reserved 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.

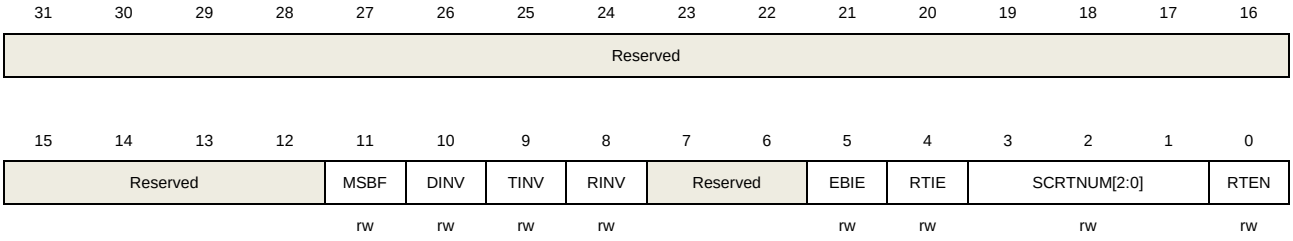
Control register 3 (USART_CTL3)

Address offset: 0x80

Reset value: 0x0000 0000

This register is not available for UART3/4.

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



Bits	Fields	Descriptions
31:12	Reserved	Must be kept the reset value.
11	MSBF	Most significant bit first This bit specifies the sequence of the data bits in transmission and reception. 0: data is transmitted/received with the LSB first. 1: data is transmitted/received with the MSB first. This bit field cannot be written when the USART is enabled (UEN=1).
10	DINV	Data bit level inversion This bit specifies the polarity of the data bits in transmission and reception. 0: Data bit signal values are not inverted. 1: Data bit signal values are inverted. This bit field cannot be written when the USART is enabled (UEN=1).
9	TINV	TX pin level inversion This bit specifies the polarity of the TX pin. 0: TX pin signal values are not inverted. 1: TX pin signal values are inverted.

		This bit field cannot be written when the USART is enabled (UEN=1).
8	RINV	RX pin level inversion This bit specifies the polarity of the RX pin. 0: RX pin signal values are not inverted. 1: RX pin signal values are inverted. This bit field cannot be written when the USART is enabled (UEN=1).
7:6	Reserved	Must be kept the reset value.
5	EBIE	Interrupt enable bit of end of block event If this bit is set, an interrupt occurs when the EBF bit in USARTx_STAT1 is set. 0: End of block interrupt is disabled. 1: End of block interrupt is enabled.
4	RTIE	Interrupt enable bit of receive timeout event If this bit is set, an interrupt occurs when the RTF bit in USART_STAT1 is set. 0: Receive timeout interrupt is disabled. 1: Receive timeout interrupt is enabled.
3:1	SCRTNUM[2:0]	Smartcard auto-retry number In Smartcard mode, these bits specify the number of retries in transmission and reception. In transmission mode, a frame can be retransmitted by SCRTNUM times. If the frame is NACKed by (SCRTNUM+1) times, the FERR is set. In reception mode, a frame reception can be tried by (SCRTNUM+1) times. If the parity bit mismatch event occurs (SCRTNUM+1) times for a frame, the RBNE and PERR bits are set. When these bits are configured as 0x0, there will be no automatic retransmission in transmit mode.
0	RTEN	Receiver timeout enable This bit enables the receive timeout counter of the USART. 0: Receiver timeout function disabled. 1: Receiver timeout function enabled.

Receiver timeout register (USART_RT)

Address offset: 0x84

Reset value: 0x0000 0000

This register is not available for UART3/4.

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

RT[15:0]
rw

Bits	Fields	Descriptions
31:24	BL[7:0]	Block Length These bits specify the block length in Smartcard T=1 Reception. Its value equals to the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1. This value, which must be programmed only once per received block, can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). The block length counter is reset when TBE=0 in Smartcard mode. In other modes, when REN=0 (receiver disabled), or when the EBF bit of USART_STAT1 is written to 0, the Block length counter is reset.
23:0	RT[23:0]	Receiver timeout threshold These bits are used to specify receiver timeout value in terms of number of baud clocks. If Smartcard mode is not enabled, the RTF bit of USART_STAT1 is set if no new start bit is detected longer than RT bits time after the last received character. If Smartcard mode is enabled, the CWT and BWT are implemented by this value. In this case, the timeout measurement is started from the start bit of the last received character. These bits can be written on the fly. The RTF flag will be set if the new value is lower than or equal to the internal timeout counter. These bits must only be programmed once per received character.

Status register 1 (USART_STAT1)

Address offset: 0x88

Reset value: 0x0000 0000

This register is not available for UART3/4.

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															BSY
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			EBF	RTF	Reserved										
			w0	w0											

Bits	Fields	Descriptions
31:17	Reserved	Must be kept the reset value.

16	BSY	Busy flag This bit is set when the USART is receiving a data frame. 0: USART reception path is idle. 1: USART reception path is working.
15:13	Reserved	Must be kept the reset value.
12	EBF	End of block flag This bit is set when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4. An interrupt occurs if the EBIE bit in USART_CTL3 is set. Software can clear this bit by writing 0 to it. 0: End of Block event does not occur. 1: End of Block event has occurred.
11	RTF	Receiver timeout flag This bit is set when the RX pin is in idle state for longer than RT bits time. An interrupt occurs if the RTIE bit in USART_CTL3 is set. Software can clear this bit by writing 0 to it. 0: Receiver timeout event does not occur. 1: Receiver timeout event has occurred.
10:0	Reserved	Must be kept the reset value.

GD control register (USART_GDCTL)

Address offset: 0xD0

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															CDIE
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CD	Reserved						CDEN	Reserved
							w0c							rw	

Bits	Fields	Descriptions
31:17	Reserved	Forced by hardware to 0.
16	CDIE	Collision detected interrupt enable. 0: Collision detected interrupt disable. 1: Collision detected interrupt enable.
15:9	Reserved	Forced by hardware to 0.
8	CD	Collision detected status

		0: no collision detected. 1: collision detected in halfduplex mode.
7:2	Reserved	Forced by hardware to 0.
1	CDEN	Collision detection enable 0: disable 1: enable
0	Reserved	Forced by hardware to 0.

20.2. Universal synchronous/asynchronous receiver /transmitter (USARTx, x=5)

20.2.1. 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 UCLK(PCLK2, CK_USART5) to produce a dedicated wide range baud rate clock 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. The data frame can be transferred from LSB or MSB bit. The polarity of the TX/RX pins can be configured independently and flexibly.

The USART support DMA function for high-speed data communication.

20.2.2. Characteristics

- NRZ standard format
- Asynchronous, full duplex communication
- Half duplex single wire communications
- Receive FIFO function
- Dual clock domain:
  - Asynchronous pclk and USART clock
  - Baud rate programming independent from the PCLK reprogramming
- Programmable baud-rate generator allowing speed up to 22.5 MBits/s when the clock frequency is 180 MHz and oversampling is by 8
- Fully programmable serial interface characteristics:
  - A data word (8 or 9 bits) LSB or MSB first
  - Even, odd or no-parity bit generation/detection
  - 0.5, 1, 1.5 or 2 stop bit generation
- Swappable Tx/Rx pin
- Configurable data polarity
- Configurable multibuffer communication using centralized DMA
- Separate enable bits for transmitter and receiver
- Parity control
  - Transmits parity bit
  - Checks parity of received data byte
- 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)
  - Block mode (T=1)
  - Direct and inverse convention
- Multiprocessor communication
  - Enter into mute mode if address match does not occur
  - Wake up from mute mode by idle line or address match detection
- Support for ModBus communication
  - Timeout feature
  - CR/LF character recognition
- Wake up from deep-sleep mode, Deep-sleep 1 mode and Deep-sleep 2 mode.
  - By standard RBNE interrupt
  - By WUF interrupt
- Various status flags
  - Flags for transfer detection: Receive buffer not empty (RBNE), receive FIFO full (RFF), 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 LIN mode: LIN break detected (LBDF)
  - Flag for multiprocessor communication: IDLE frame detected (IDLEF)
  - Flag for ModBus communication: address/character match (AMF) and receiver timeout (RTF)
  - Flags for smartcard block mode: end of block (EBF) and receiver timeout (RTF)
  - Wakeup from deep-sleep mode flag
  - Interrupt occurs at these events when the corresponding interrupt enable bits are set

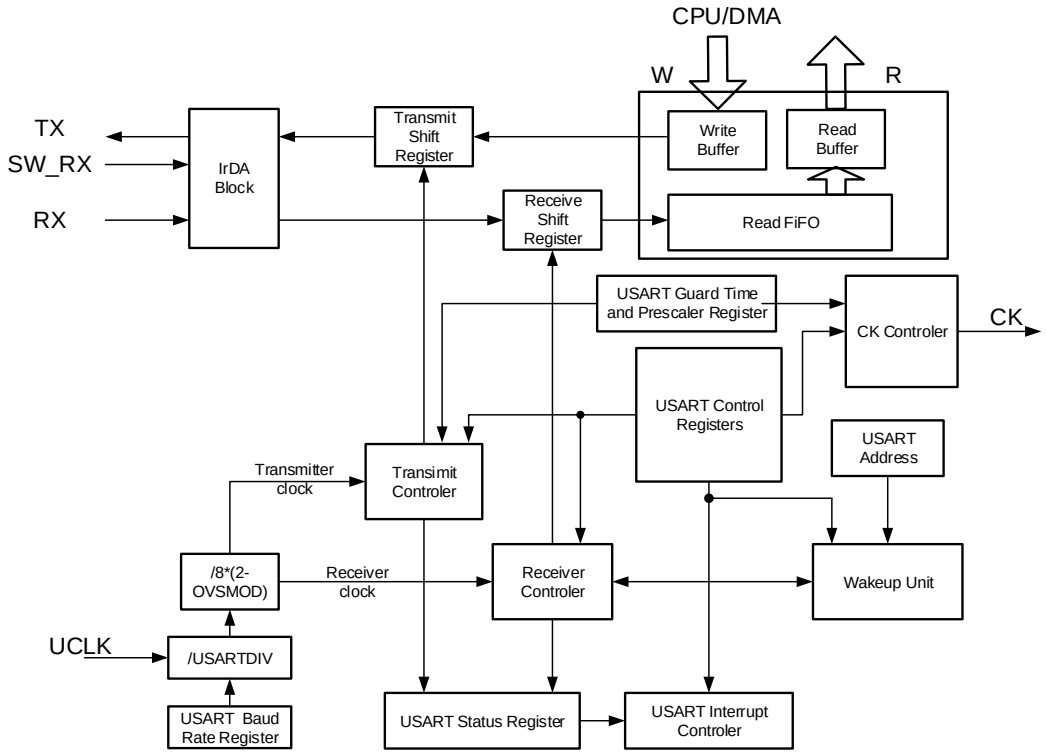
20.2.3. Function overview

The interface is externally connected to another device by the main pins listed in ***Table 20-4. Description of USART important pins***.

Table 20-4. Description of USART important pins

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

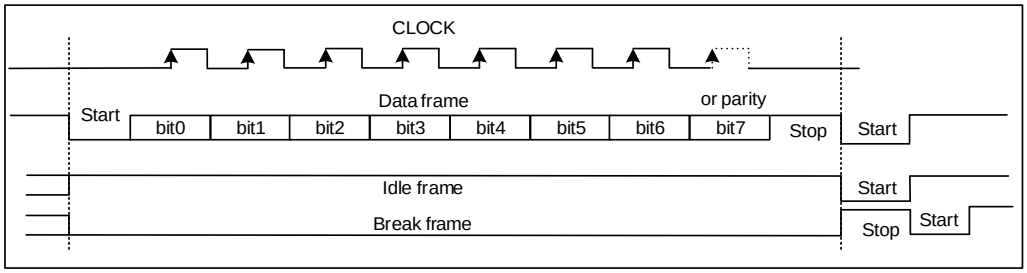
Figure 20-17. USART module block diagram



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 20-18. 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 20-5. Configuration of stop bits

STB[1:0]	stop bit length (bit)	usage description
00	1	Default value

STB[1:0]	stop bit length (bit)	usage description
01	0.5	Smartcard mode for receiving
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.

The break frame structure is a 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 UCLK, the configuration of the baud rate generator and the oversampling mode.

Baud rate generation

The baud-rate divider is a 16-bit number which consists 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 with the UCLK:

In case of oversampling by 16, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{16 \times \text{Baud Rate}} \quad (20\text{-}3)$$

In case of oversampling by 8, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{8 \times \text{Baud Rate}} \quad (20\text{-}4)$$

For example, when oversampled by 16:

1. Get USARTDIV by caculating the value of USART_BUAD:
   If USART_BUAD=0x21D, then INTDIV=33 (0x21), FRADIV=13 (0xD).
   USARTDIV=33+13/16=33.81.

2. Get the value of USART_BUAD by calculating the value of USARTDIV:
   If USARTDIV=30.37, then INTDIV=30 (0x1E).
   16*0.37=5.92, the nearest integer is 6, so FRADIV=6 (0x6).
   USART_BUAD=0x1E6.

   Note: If the roundness of FRADIV is 16 (overflow), the carry must be added to the integer part.

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. The polarity of the TX pin can be configured by the TINV bit in the USART_CTL1 register. Clock pulses can output through the CK pin.

After the TEN bit is set, an idle frame will be sent. The TEN bit should not be cleared while the transmission is ongoing.

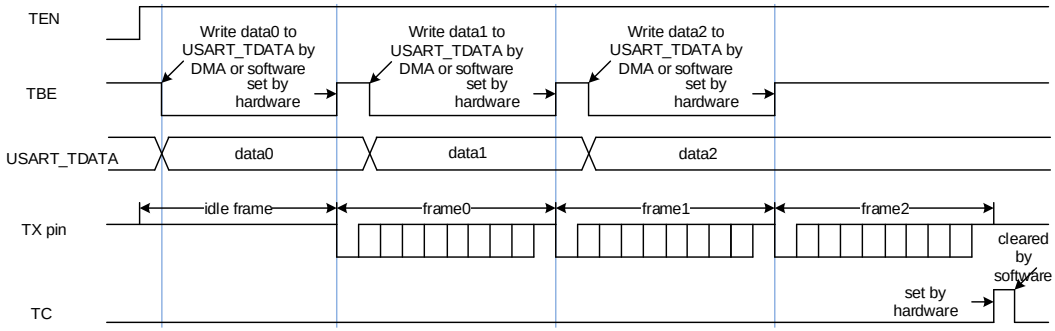
After power on, the TBE bit is high by default. Data can be written to the USART_TDATA when the TBE bit in the USART_STAT register is asserted. The TBE bit is cleared by writing USART_TDATA register and it is set by hardware after the data is put into the transmit shift register. If a data is written to the USART_TDATA 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_TDATA 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 will be generated if the corresponding interrupt enable bit (TCIE) is set in the USART_CTL0 register.

The USART transmit procedure is shown in ***Figure 20-19. USART transmit procedure***. The software operating process is as follows:

1. Write the WL bit in USART_CTL0 to set the data bits length.
2. Set the STB[1:0] bits in USART_CTL1 to configure the number of stop bits.
3. Enable DMA (DENT bit) in USART_CTL2 if multibuffer communication is selected.
4. Set the baud rate in USART_BAUD.
5. Set the UEN bit in USART_CTL0 to enable the USART.
6. Set the TEN bit in USART_CTL0.
7. Wait for the TBE being asserted.
8. Write the data to the USART_TDATA register.
9. Repeat step7-8 for each data, if DMA is not enabled.
10. Wait until TC=1 to finish.

Figure 20-19. USART transmit procedure



It is necessary to wait for the TC bit to be asserted before disabling the USART or entering the power saving mode. The TC bit can be cleared by writing 1 to TCC bit in USART_INTC register.

The break frame is sent when the SBKCMD bit is set, and SBKCMD bit is reset after the transmission.

USART receiver

After power on, the USART receiver can be enabled by the following procedure:

1. Write the WL bit in USART_CTL0 to set the data bits length.
2. Set the STB[1:0] bits in USART_CTL1.
3. Enable DMA (DENR bit) in USART_CTL2 if multibuffer communication is selected.
4. Set the baud rate in USART_BAUD.
5. Set the UEN bit in USART_CTL0 to enable the USART.
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.

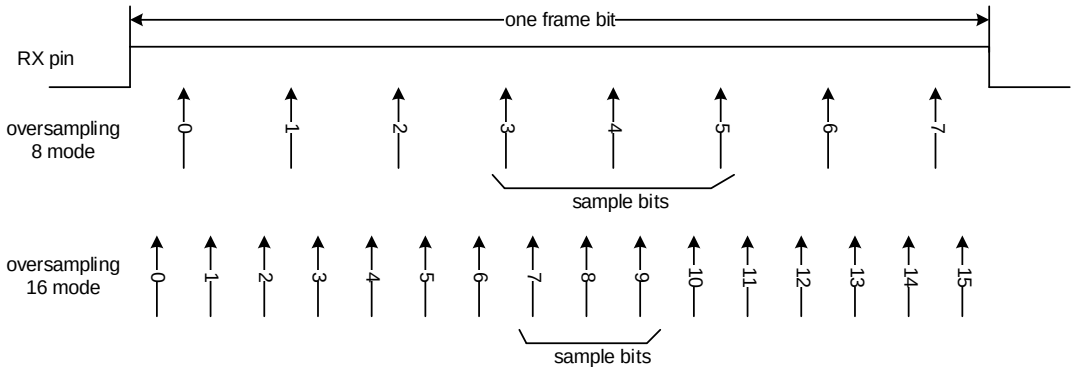
When a frame is received, the RBNE bit in USART_STAT is asserted, an interrupt is generated if the corresponding interrupt enable bit (RBNEIE) is set in the USART_CTL0 register. The status of the reception are stored in the USART_STAT register.

The software can get the received data by reading the USART_RDATA register directly, or through DMA. The RBNE bit is cleared by a read operation on the USART_RDATA 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. If the oversampling 8 mode is enabled, the 3rd, 4th and 5th samples are used, 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 value of the three samples of any bit 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 will be generated, If the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set. If the OSB bit in USART_CTL2 register is set, the receiver gets only one sample to evaluate a bit value. In this situation, no noisy error will be detected.

Figure 20-20. Oversampling method of a receive frame bit (OSB=0)



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.

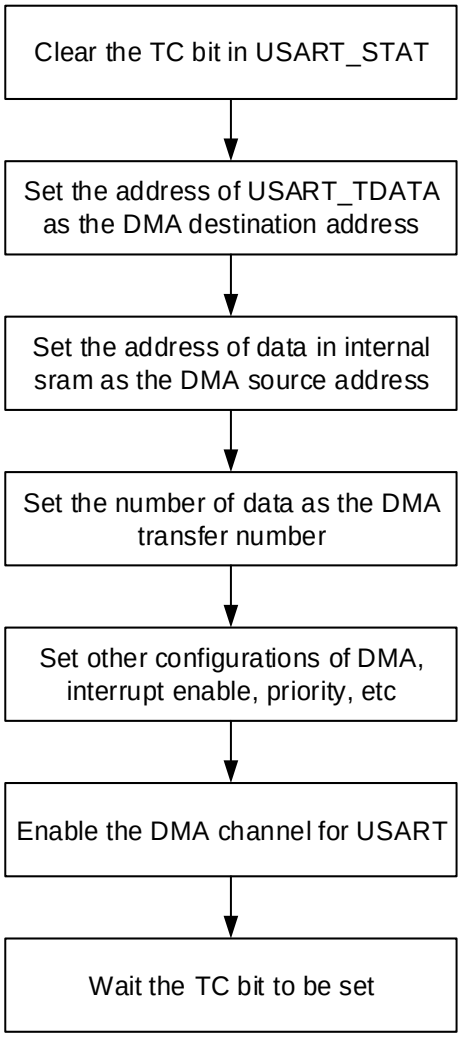
If a noise error (NERR), parity error (PERR), frame error (FERR) or overrun error (ORERR) occurs during reception, NERR, PERR, FERR or ORERR will be set at the same time with RBNE. If the receive DMA is not enabled, when the RBNE interrupt occurs, software need to check whether there is a noise error, parity error, frame error or overrun error.

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 are shown in ***Figure 20-21. Configuration step when using DMA for USART transmission***.

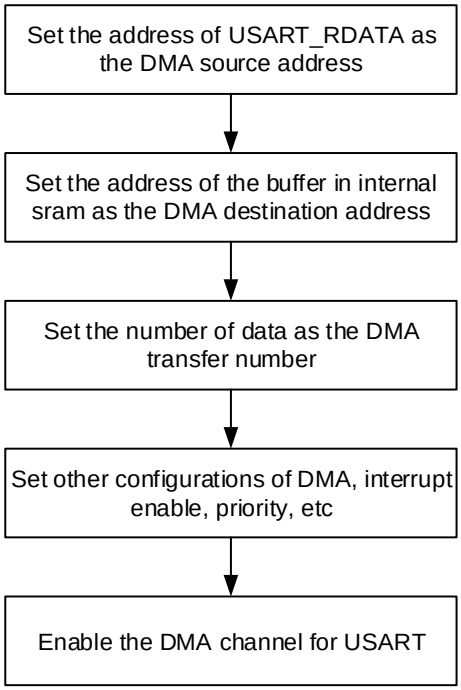
Figure 20-21. Configuration step when using DMA for USART transmission



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 steps are shown in ***Figure 20-22. 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 20-22. Configuration step when using DMA for USART reception



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.

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 writing 1 to the MMCMD bit in USART_CMD register.

If a USART is in mute mode, all of the receive status bits cannot be set. 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 it is woken up by an idle frame, the IDLEF bit in USART_STAT will not be 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 or 7 bits, which are configured by the ADDM bit of the USART_CTL1 register, of an address frame is the same as the ADDR bits in the USART_CTL1 register, the hardware will clear the RWU bit and exits the mute mode. The RBNE bit will be set when the frame that wakes up the USART. The status bits are available in the USART_STAT register. If the LSB 4/7 bits of an address frame defers from the ADDR 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 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 bit. If the ADDM bit is set and the receive frame is a 7bit data, the LSB 6 bits will be compared with ADDR[5:0]. If the ADDM bit is set and the receive frame is a 9bit data, the LSB 8 bits will be compared with ADDR[7:0].

LIN mode

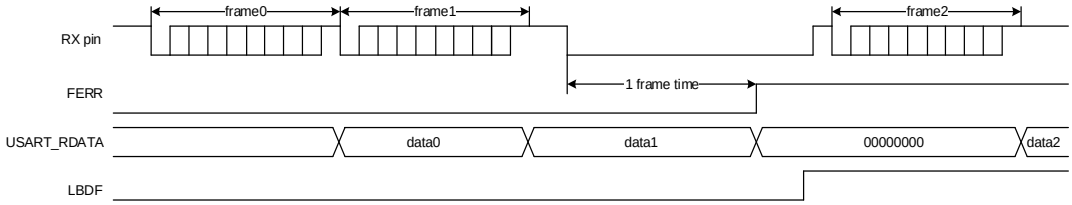
The local interconnection network mode is enabled by setting the LMEN bit in USART_CTL1. The CKEN, STB[1:0] bit in USART_CTL1 and the SCEN, HDEN, IREN bits in USART_CTL2 should be cleared in LIN mode.

When transmitting a normal data frame, the transmission procedure is the same as the normal USART mode. The data bits length can only be 8. And the break frame is 13-bit ’0’, followed by 1 stop bit.

The break detection function is totally independent of 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 configuring 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 bit in USART_STAT is set. An interrupt occurs if the LBDIE bit in USART_CTL1 is set.

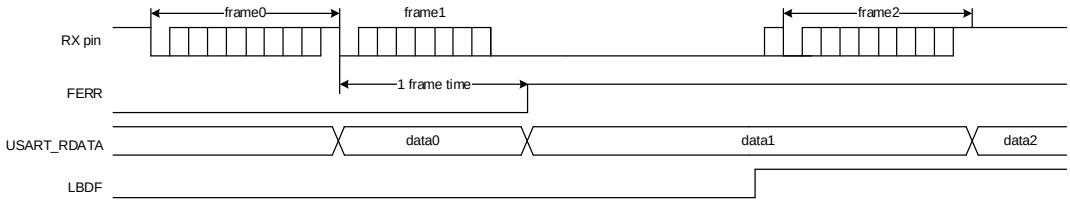
As shown in ***Figure 20-23. Break frame occurs during idle state***, if a break frame occurs during the idle state on the RX pin, the USART receiver will receive an all ‘0’ frame, with an asserted FERR status.

Figure 20-23. Break frame occurs during idle state



As shown in ***Figure 20-24. 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 20-24. Break frame occurs during a frame



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 cleared in synchronous mode. The CK pin is

the clock output of the synchronous USART transmitter, and can be only activated when the TEN bit is enabled. No clock pulse will be sent through the CK pin during the transmission of the start bit and stop bit. 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 clock output is also not activated during idle and break frame sending. 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).

The clock is synchronized with the data transmitted. The receiver in synchronous mode samples the data on the transmitter clock without any oversampling.

Figure 20-25. Example of USART in synchronous mode

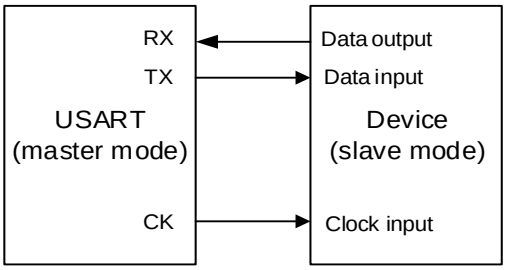
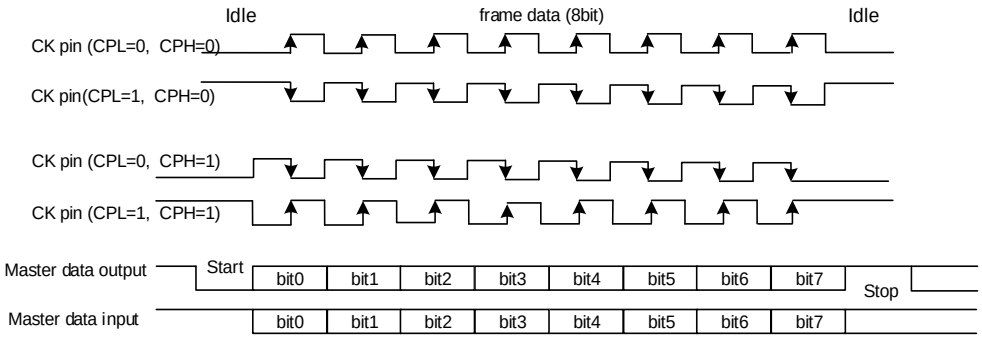


Figure 20-26. 8-bit format USART synchronous waveform (CLEN=1)

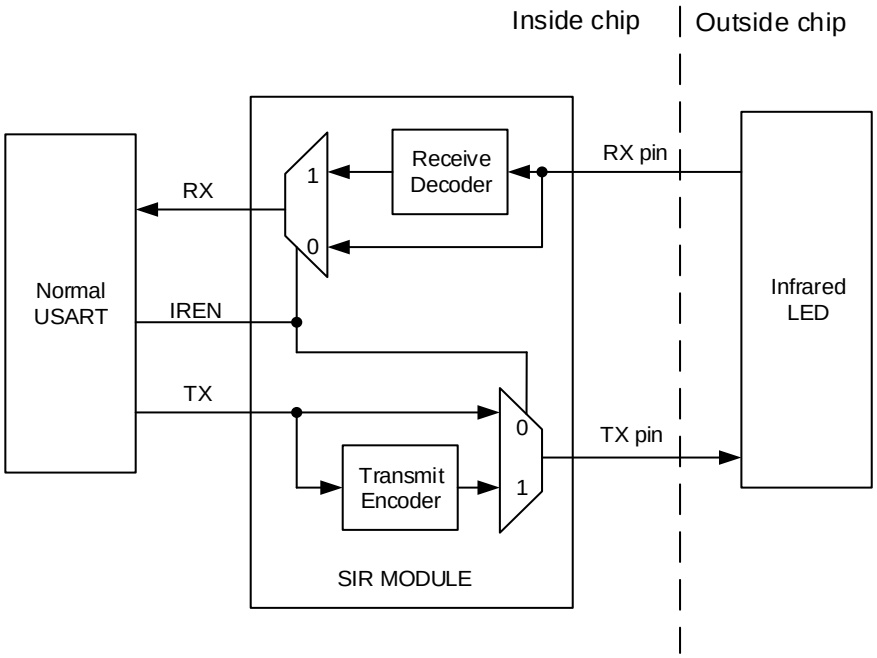


IrDA SIR ENDEC mode

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 cleared 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.

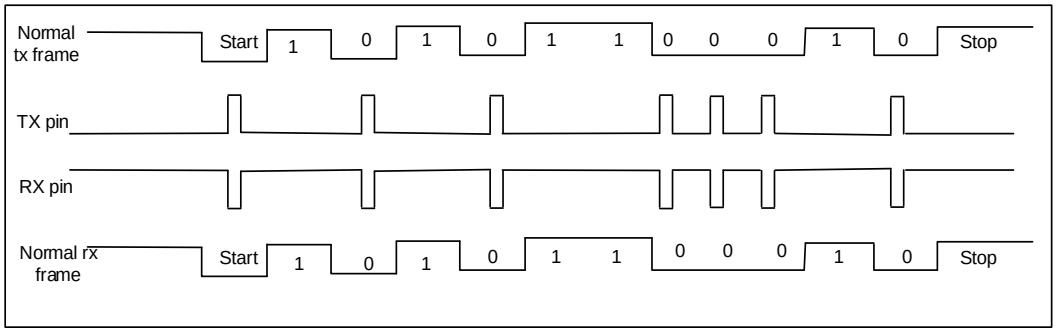
Figure 20-27. IrDA SIR ENDEC module



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 of 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 20-28. IrDA data modulation



The SIR sub module 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 division 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 period. The receiver decoder works in the same manner as the normal IrDA mode.

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 cleared in half-duplex communication mode.

Only one wire is used in half-duplex mode. The TX and RX pins are connected together internally. The TX pin should be configured as IO pin. The conflicts should be controlled by the software. When the TEN bit is set, the data in the data register will be sent.

Smartcard (ISO7816-3) mode

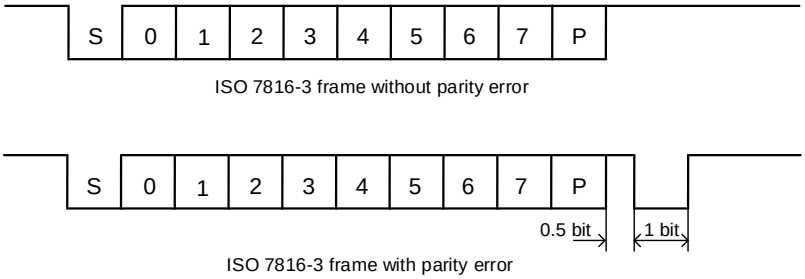
The smartcard mode is an asynchronous mode, which is designed to support the ISO7816-3 protocol. Both the character (T=0) mode and the block (T=1) mode are supported. The smartcard mode is enabled by setting the SCEN bit in USART_CTL2. The LMEN bit in USART_CTL1 and HDEN, IREN bits in USART_CTL2 should be reset in smartcard mode.

A clock is provided to the smartcard if the CKEN bit is set. The clock can be divided for other use.

The frame consists of 1 start bit, 9 data bits (1 parity bit included) and 1.5 stop bits.

The smartcard mode is a half-duplex communication protocol. When connected to a smartcard, the TX pin must be configured as open drain mode, and drives a bidirectional line that is also driven by the smartcard.

Figure 20-29. ISO7816-3 frame format



Character (T=0) mode

Compared 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 USART5_GP. In Smartcard mode, the internal guard time counter starts counting 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 for SCRTNUM times. An interframe

gap of 2.5 bits time will be inserted before the start of a resented frame. At the end of the last repeated character the TC bit is set immediately without guard time. The USART will stop transmitting and assert the frame error status if it still receives the NACK signal after the programmed number of retries. 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 smartcard. Then a frame error occurs in smartcard side. The RBNE/receive DMA request is not activated if the received character is erroneous. According to the protocol, the smartcard can resend the data. The USART stops transmitting the NACK and the error is regarded as a parity error if the received character is still erroneous after the maximum number of retries which is specified in the SCRTNUM bit field. The NACK signal is enabled by setting the NKEN bit in USART_CTL2.

The idle frame and break frame are not supported in the Smartcard mode.

Block (T=1) mode

In block (T=1) mode, the NKEN bit in the USART_CTL2 register should be cleared to deactivate the NACK transmission.

When requesting a read from the smartcard, the RT[23:0] bits in USART_RT register should be programmed with the BWT (block wait time) - 11 value and RBNEIE must be set. A timeout interrupt will be generated, if no answer is received from the card before the expiration of this period. If the first character is received before the expiration of the period, it is signaled by the RBNE interrupt. If DMA is used to read from the smartcard in block mode, the DMA must be enabled only after the first character is received.

In order to allow the automatic check of the maximum wait time between two consecutive characters, the USART_RT register must be programmed to the CWT (character wait time) - 11 value, which is expressed in baudtime units, after the reception of the first character (RBNE interrupt). The USART signals to the software through the RT flag and interrupt (when RTIE bit is set), if the smartcard doesn’t send a new character in less than the CWT period after the end of the previous character.

The USART uses a block length counter, which is reset when the USART is transmitting (TBE=0), to count the number of received characters. The length of the block, which must be programmed in the BL[7:0] bits in the USART_RT register, is received from the smartcard in the third byte of the block (prologue field). This register field must be programmed to the minimum value (0x0), before the start of the block, when using DMA mode. With this value, an interrupt is generated after the 4th received character. The software must read the third byte as block length from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BL value. However, before the start of the block, the maximum value of BL (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

The total block length (including prologue, epilogue and information fields) equals BL+4. The end of the block is signaled to the software through the EBF flag and interrupt (when EBIE bit is set). The RT interrupt may occur in case of an error in the block length.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to high state of the line and parity is even. In this case, the following control bits must be programmed: MSBF=0, DINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an low state on the signal line and parity is even. In this case, the following control bits must be programmed: MSBF=1, DINV=1.

ModBus communication

The USART offers basic support for the implementation of ModBus/RTU and ModBus/ASCII protocols by implementing an end of block detection.

In the ModBus/RTU mode, the end of one block is recognized by an idle line for more than 2 characters time. This function is implemented through the programmable timeout function.

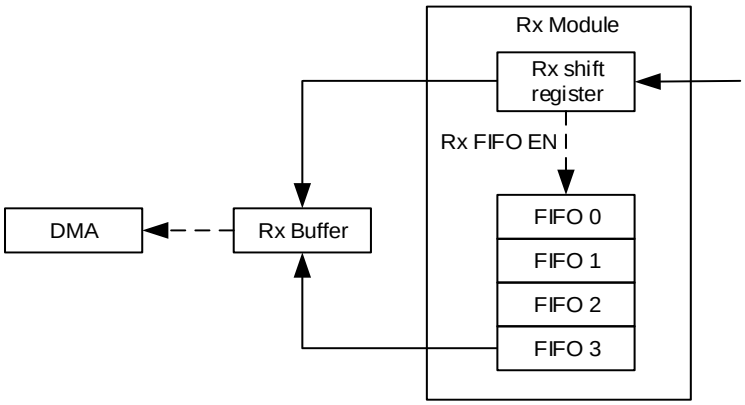
To detect the idle line, the RTEN bit in the USART_CTL1 register and the RTIE in the USART_CTL0 register must be set. The USART_RT register must be set to the value corresponding to a timeout of 2 characters time. After the last stop bit is received, when the receive line is idle for this duration, an interrupt will be generated, informing the software that the current block reception is completed.

In the ModBus/ASCII mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function by programming the LF ASCII code in the ADDR field and activating the address match interrupt (AMIE=1). When a LF has been received or can check the CR/LF in the DMA buffer, the software will be informed.

Receive FIFO

The receive FIFO can be enabled by setting the RFEN bit of the USART_RFCS register to avoid the overrun error when the CPU can’t serve the RBNE interrupt immediately. Up to 5 frames receive data can be stored in the receive FIFO and receive buffer. The RFFINT flag will be set when the receive FIFO is full. An interrupt is generated if the RFFIE bit is set.

Figure 20-30. USART receive FIFO structure



If the software read receive data buffer in the routing of the RBNE interrupt, the RBNEIE bit should be reset at the beginning of the routing and set after all of the receive data is read out. The PERR/NERR/FERR/EBF flags should be cleared before reading a receive data out.

Wakeup from deep-sleep mode

The USART is able to wake up the MCU from deep-sleep mode, deep-sleep 1 mode and deep-sleep 2 mode by the standard RBNE interrupt or the WUM interrupt.

The UESM bit must be set and the USART clock must be set to IRC8M or LXTAL (refer to the reset and clock unit RCU section).

When using the standard RBNE interrupt, the RBNEIE bit must be set before entering deep-sleep mode, deep-sleep 1 mode and deep-sleep 2.

When using the WUIE interrupt, the source of WUIE interrupt may be selected through the WUM bit fields.

DMA must be disabled before entering deep-sleep mode, deep-sleep 1 mode and deep-sleep 2. Before entering deep-sleep mode, software must check that the USART is not performing a transfer, by checking the BSY flag in the USART_STAT register. The REA bit must be checked to ensure the USART is actually enabled.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUIE bit is set, independently of whether the MCU is in stop or active mode.

USART interrupts

The USART interrupt events and flags are listed in ***Table 20-6. USART interrupt requests***.

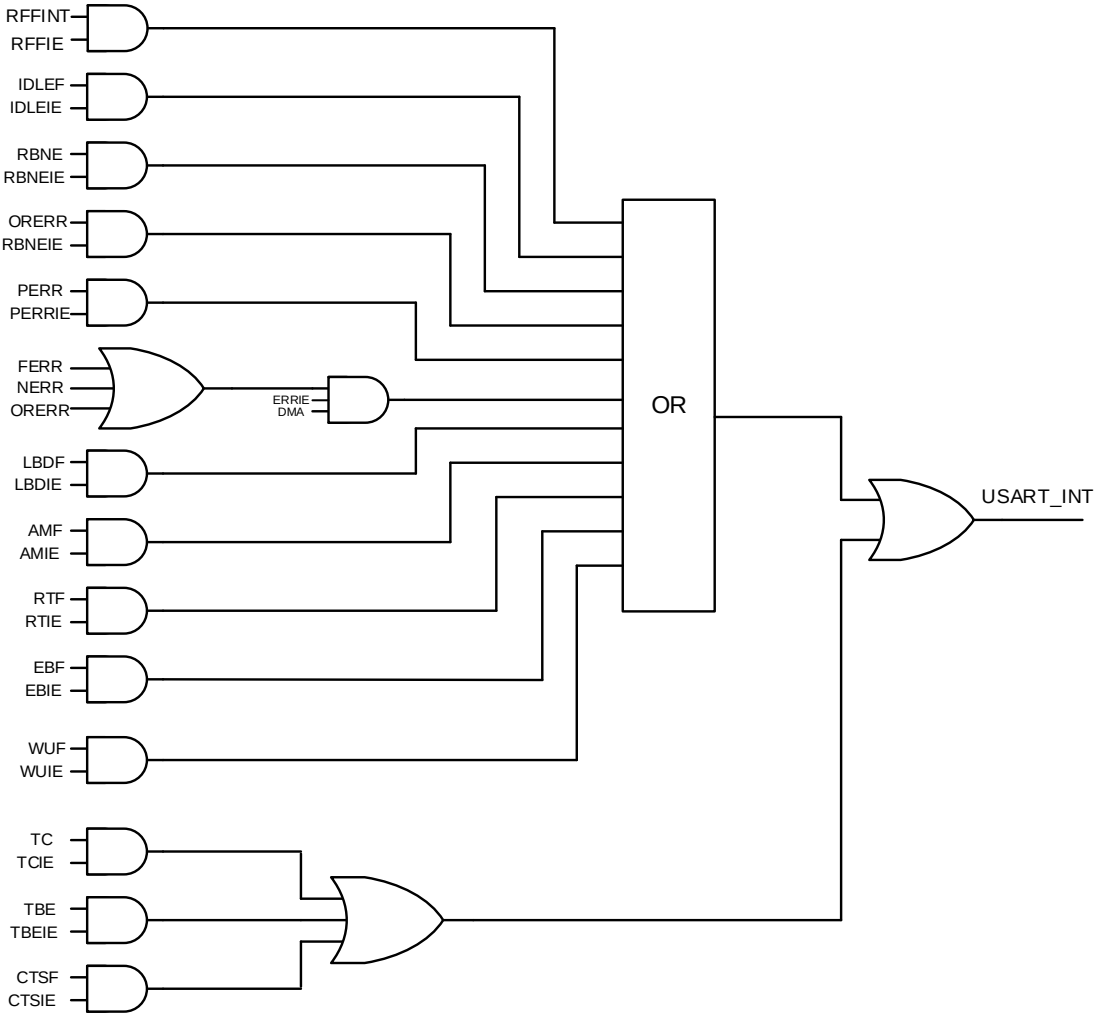
Table 20-6. USART interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TBE	TBEIE
Transmission complete	TC	TCIE
Received data ready to be	RBNE	RBNEIE

Interrupt event	Event flag	Enable Control bit
read		
Overrun error detected	ORERR	
Receive FIFO full	RFFINT	RFFIE
Idle line detected	IDLEF	IDLEIE
Parity error flag	PERR	PERRIE
Break detected flag in LIN mode	LBDF	LBDIE
Reception errors (noise flag, overrun error, framing error) in DMA reception	NERR or ORERR or FERR	ERRIE
Character match	AMF	AMIE
Receiver timeout error	RTF	RTIE
End of block	EBF	EBIE
Wakeup from deep-sleep mode	WUF	WUIE

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 20-31. USART interrupt mapping diagram



20.2.4. Register defintion

USART5 base address: 0x4001 7000

Control register 0 (USART_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				EBIE	RTIE	Reserved									
				rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVSMOD	AMIE	MEN	WL	WM	PCEN	PM	PERRIE	TBEIE	TCIE	RBNEIE	IDLEIE	TEN	REN	UESM	UEN
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	EBIE	End of Block interrupt enable. 0: End of Block interrupt is disabled. 1: End of Block interrupt is enabled.
26	RTIE	Receiver timeout interrupt enable. 0: Receiver timeout interrupt is disabled. 1: Receiver timeout interrupt is enabled.
25:16	Reserved	Must be kept at reset value.
15	OVSMOD	Oversample mode 0: Oversampling by 16. 1: Oversampling by 8. This bit must be kept cleared in LIN, IrDA and smartcard modes. This bit field cannot be written when the USART is enabled (UEN=1).
14	AMIE	ADDR match interrupt enable. 0: ADDR match interrupt is disabled. 1: ADDR match interrupt is enabled.
13	MEN	Mute mode enable. 0: Mute mode disabled. 1: Mute mode enabled.
12	WL	Word length. 0: 8 Data bits 1: 9 Data bits

		This bit field cannot be written when the USART is enabled (UEN=1).
11	WM	Wakeup method in mute mode 0: Idle Line 1: Address Mark This bit field cannot be written when the USART is enabled (UEN=1).
10	PCEN	Parity control enable. 0: Parity control disabled. 1: Parity control enabled. This bit field cannot be written when the USART is enabled (UEN=1).
9	PM	Parity mode 0: Even parity 1: Odd parity This bit field cannot be written when the USART is enabled (UEN=1).
8	PERRIE	Parity error interrupt enable. 0: Parity error interrupt is disabled. 1: An interrupt will occur whenever the PERR bit is set in USART_STAT.
7	TBEIE	Transmitter register empty interrupt enable. 0: Interrupt is inhibited. 1: An interrupt will occur whenever the TBE bit is set in USART_STAT.
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. 0: Read data register not empty interrupt and overrun error interrupt disabled. 1: An interrupt will occur whenever the ORERR bit is set or the RBNE bit is set in USART_STAT.
4	IDLEIE	IDLE line detected interrupt enable. 0: IDLE line detected interrupt disabled. 1: An interrupt will occur whenever the IDLEF bit is set in USART_STAT.
3	TEN	Transmitter enable. 0: Transmitter is disabled. 1: Transmitter is enabled.
2	REN	Receiver enable. 0: Receiver is disabled. 1: Receiver is enabled and begins searching for a start bit.
1	UESM	USART enable in Deep-sleep mod. 0: USART not able to wake up the MCU from Deep-sleep mode.

1: USART able to wake up the MCU from Deep-sleep mode. Providing that the clock source for the USART must be IRC8M or LXTAL.

0 UEN USART enable
0: USART prescaler and outputs disabled.
1: USART prescaler and outputs enabled.

Control register 1 (USART_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
ADDR[7:0]								RTEN	Reserved			MSBF	DINV	TINV	RINV
rw								rw				rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STRP	LMEN	STB[1:0]		CKEN	CPL	CPH	CLEN	Reserved	LBDIE	LBLEN	ADDM	Reserved			
rw	rw	rw		rw	rw	rw	rw		rw	rw	rw				

Bits	Fields	Descriptions
31:24	ADDR[7:0]	Address of the USART terminal. These bits give the address of the USART terminal. In multiprocessor communication during mute mode or deep-sleep mode, this is used for wakeup with address match detection. The received frame, the MSB of which is equal to 1, will be compared to these bits. When the ADDM bit is reset, only the ADDR[3:0] bits are used to compare. In normal reception, these bits are also used for character detection. The whole received character (8-bit) is compared to the ADDR[7:0] value and AMF flag is set on matching. This bit field cannot be written when both reception (REN=1) and USART (UEN=1) are enabled.
23	RTEN	Receiver timeout enable 0: Receiver timeout function disabled. 1: Receiver timeout function enabled.
22:20	Reserved	Must be kept at reset value.
19	MSBF	Most significant bit first. 0: Data is transmitted/received with the LSB first. 1: Data is transmitted/received with the MSB first. This bit field cannot be written when the USART is enabled (UEN=1).
18	DINV	Data bit level inversion. 0: Data bit signal values are not inverted.

		1: Data bit signal values are inverted. This bit field cannot be written when the USART is enabled (UEN=1).
17	TINV	TX pin level inversion. 0: TX pin signal values are not inverted. 1: TX pin signal values are inverted. This bit field cannot be written when the USART is enabled (UEN=1).
16	RINV	RX pin level inversion. 0: RX pin signal values are not inverted. 1: RX pin signal values are inverted. This bit field cannot be written when the USART is enabled (UEN=1).
15	STRP	Swap TX/RX pins. 0: The TX and RX pins functions are not swapped. 1: The TX and RX pins functions are swapped. This bit field cannot be written when the USART is enabled (UEN=1).
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).
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).
10	CPL	Clock polarity 0: Steady low value on CK pin outside transmission window in synchronous mode. 1: Steady high value on CK pin outside transmission window in synchronous mode. This bit field cannot be written when the USART is enabled (UEN=1).
9	CPH	Clock phase 0: The first clock transition is the first data capture edge in synchronous mode. 1: The second clock transition is the first data capture edge in synchronous mode. This bit field cannot be written when the USART is enabled (UEN=1).
8	CLEN	CK length 0: The clock pulse of the last data bit (MSB) is not output to the CK pin in synchronous mode.

		1: The clock pulse of the last data bit (MSB) is output to the CK pin in synchronous mode. This bit field cannot be written when the USART is enabled (UEN=1).
7	Reserved	Must be kept at reset value.
6	LBDIE	LIN break detection interrupt enable. 0: LIN break detection interrupt is disabled. 1: An interrupt will occur whenever the LBDF bit is set in USART_STAT.
5	LBLEN	LIN break frame length 0: 10 bit break detection 1: 11 bit break detection This bit field cannot be written when the USART is enabled (UEN=1).
4	ADDM	Address detection mode. This bit is used to select between 4-bit address detection and full-bit address detection. 0: 4-bit address detection. 1: full-bit address detection. In 7-bit, 8-bit and 9-bit data modes, the address detection is done on 6-bit, 7-bit and 8-bit address (ADDR[5:0], ADDR[6:0] and ADDR[7:0]) respectively. This bit field cannot be written when the USART is enabled (UEN=1).
3:0	Reserved	Must be kept at reset value.

Control register 2 (USART_CTL2)

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									WUIE	WUM[1:0]		SCRTNUM[2:0]			Reserved
									rw	rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DDRE	OVRD	OSB	Reserved			DENT	DENR	SCEN	NKEN	HDEN	IRLP	IREN	ERRIE
		rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value.
22	WUIE	Wakeup from Deep-sleep mode interrupt enable. 0: Wakeup from Deep-sleep mode interrupt is disabled. 1: Wakeup from Deep-sleep mode interrupt is enabled.
21:20	WUM[1:0]	Wakeup mode from Deep-sleep mode.

		These bits are used to specify the event which activates the WUF (Wakeup from Deep-sleep mode flag) in the USART_STAT register. 00: WUF active on address match, which is defined by ADDR and ADDM. 01: Reserved. 10: WUF active on Start bit. 11: WUF active on RBNE. This bit field cannot be written when the USART is enabled (UEN=1).
19:17	SCRTNUM[2:0]	Smartcard auto-retry number In smartcard mode, these bits specify the number of retries in transmission and reception. In transmission mode, a transmission error (FERR bit set) will occur after this number of automatic retransmission retries. In reception mode, reception error (RBNE and PERR bits set) will occur after this number or erroneous reception trials. When these bits are configured as 0x0, there will be no automatic retransmission in transmit mode. This bit field is only can be cleared to 0 when the USART is enabled (UEN=1), to stop retransmission.
16:14	Reserved	Must be kept at reset value.
13	DDRE	Mask DMA request on reception error. 0: DMA is not disabled in case of reception error. The DMA request is not asserted to make sure the erroneous data is not transferred, but the next correct received data will be transferred. The RBNE is kept 0 to prevent overrun when reception error, but the corresponding error flag is set. This mode can be used in smartcard mode. 1: The DMA request is not asserted in case of reception error until the error flag is cleared. The RBNE flag and corresponding error flag will be set. The software must first disable the DMA request (DMAR = 0) or clear RBNE before clearing the error flag. This bit field cannot be written when the USART is enabled (UEN=1).
12	OVRD	Overrun disable 0: Overrun functionality is enabled. The ORERR error flag will be set when received data is not read before receiving new data, and the new data will be lost. 1: Overrun functionality is disabled. The ORERR error flag will not be set when received data is not read before receiving new data, and the new received data overwrites the previous content of the USART_RDATA register. This bit field cannot be written when the USART is enabled (UEN=1).
11	OSB	One sample bit method 0: Three sample bit method. 1: One sample bit method. This bit field cannot be written when the USART is enabled (UEN=1).

10:8	Reserved	Must be kept at reset value.
7	DENT	DMA enable for transmission. 0: DMA mode is disabled for transmission. 1: DMA mode is enabled for transmission.
6	DENR	DMA enable for reception. 0: DMA mode is disabled for reception. 1: DMA mode is enabled for reception.
5	SCEN	Smartcard mode enable. 0: Smartcard Mode disabled. 1: Smartcard Mode enabled. This bit field cannot be written when the USART is enabled (UEN=1).
4	NKEN	NACK enable in Smartcard mode. 0: Disable NACK transmission when parity error. 1: Enable NACK transmission when parity error. This bit field cannot be written when the USART is enabled (UEN=1).
3	HDEN	Half-duplex enable. 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 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. 0: IrDA disabled. 1: IrDA enabled. This bit field cannot be written when the USART is enabled (UEN=1).
0	ERRIE	Error interrupt enable. 0: Error interrupt disabled. 1: An interrupt will occur whenever the FERR bit or the ORERR bit or the NERR bit is set in USART_STAT in multibuffer communication.

Baud rate generator register (USART_BAUD)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).
This register cannot be written when the USART is enabled (UEN=1).

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
BRR [15:4]												BRR[3:0]			
rw												rw			

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:4	BRR[15:4]	Integer of baud-rate divider INTDIV = BRR[15:4]
3:0	BRR [3:0]	Fraction of baud-rate divider If OVSMOD = 0, FRADIV = BRR [3:0]; If OVSMOD = 1, FRADIV = BRR [2:0], BRR [3] must be reset.

Prescaler and guard time configuration register (USART_GP)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).
This register cannot be written when the USART is enabled (UEN=1).

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							

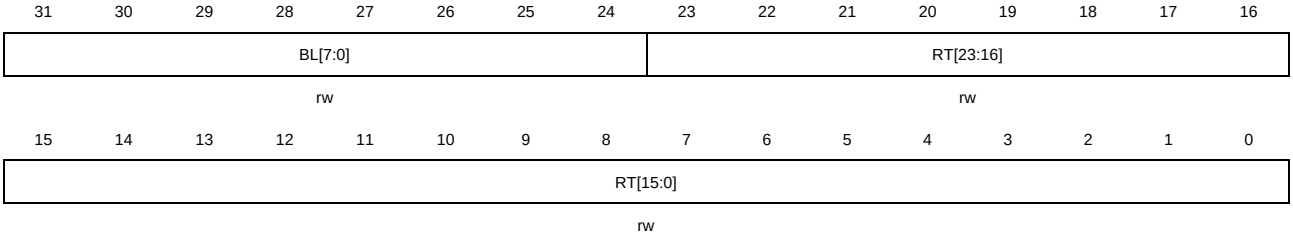
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:8	GUAT[7:0]	Guard time value in smartcard mode. This bit field cannot be written when the USART is enabled (UEN=1).
7:0	PSC[7:0]	Prescaler value for dividing the system clock. In IrDA Low-power mode, the division factor is the prescaler value. 00000000: Reserved - do not program this value. 00000001: divides the source clock by 1. 00000010: divides the source clock by 2. ... In IrDA normal mode, 00000001: can be set this value only. In smartcard mode, the prescaler value for dividing the system clock is stored in PSC[4:0] bits. And the bits of PSC[7:5] must be kept at reset value. The division

factor is twice as the prescaler value.
00000: Reserved - do not program this value.
00001: divides the source clock by 2.
00010: divides the source clock by 4.
00011: divides the source clock by 6.
...
This bit field cannot be written when the USART is enabled (UEN=1).

Receiver timeout register (USART_RT)

Address offset: 0x14
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:24	BL[7:0]	Block Length. These bits specify the block length in smartcard T=1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1. This value, which must be programmed only once per received block, can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). The block length counter is reset when TBE=0 in smartcard mode. In other modes, when REN=0 (receiver disabled) and/or when the EBC bit is written to 1, the Block length counter is reset.
23:0	RT[23:0]	Receiver timeout threshold. These bits are used to specify receiver timeout value in terms of number of baud clocks. In standard mode, the RTF flag is set if no new start bit is detected for more than the RT value after the last received character. In smartcard mode, the CWT and BWT are implemented by this value. In this case, the timeout measurement is started from the start bit of the last received character. These bits can be written on the fly. The RTF flag will be set if the new value is lower than or equal to the counter. These bits must only be programmed once per received character.

Command register (USART_CMD)

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											TXFCMD	RXFCMD	MMCMD	SBKCMD	Reserved
											w	w	w	w	

Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value.
4	TXFCMD	Transmit data flush request. Writing 1 to this bit sets the TBE flag, to discard the transmit data.
3	RXFCMD	Receive data flush command. Writing 1 to this bit clears the RBNE flag to discard the received data without reading it.
2	MMCMD	Mute mode command. Writing 1 to this bit makes the USART into mute mode and sets the RWU flag.
1	SBKCMD	Send break command. Writing 1 to this bit sets the SBKF flag and makes the USART send a BREAK frame, as soon as the transmit machine is idle.
0	Reserved	Must be kept at reset value.

Status register (USART_STAT)

Address offset: 0x1C
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									REA	TEA	WUF	RWU	SBF	AMF	BSY
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			EBF	RTF	Reserved		LBDF	TBE	TC	RBNE	IDLEF	ORERR	NERR	FERR	PERR
			r	r			r	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value.

22	REA	Receive enable acknowledge flag. This bit, which is set/reset by hardware, reflects the receive enable state of the USART core logic. 0: The USART core receiving logic has not been enabled. 1: The USART core receiving logic has been enabled.
21	TEA	Transmit enable acknowledge flag. This bit, which is set/reset by hardware, reflects the transmit enable state of the USART core logic. 0: The USART core transmitting logic has not been enabled. 1: The USART core transmitting logic has been enabled.
20	WUF	Wakeup from deep-sleep mode flag. 0: No wakeup from deep-sleep mode. 1: Wakeup from Deep-sleep mode. An interrupt is generated if WUFIE=1 in the USART_CTL2 register and the MCU is in Deep-sleep mode. This bit is set by hardware when a wakeup event, which is defined by the WUM bit field, is detected. Cleared by writing a 1 to the WUC in the USART_INTC register. This bit can also be cleared when UESM is cleared.
19	RWU	Receiver wakeup from mute mode. This bit is used to indicate if the USART is in mute mode. 0: Receiver in active mode. 1: Receiver in mute mode. It is cleared/set by hardware when a wakeup/mute sequence (address or IDLEIE) is recognized, which is selected by the WAKE bit in the USART_CTL0 register. This bit can only be set by writing 1 to the MMCMD bit in the USART_CMD register when wakeup on IDLEIE mode is selected.
18	SBF	Send break flag. 0: No break character is transmitted. 1: Break character will be transmitted. This bit indicates that a send break character was requested. Set by software, by writing 1 to the SBKCMD bit in the USART_CMD register. Cleared by hardware during the stop bit of break transmission.
17	AMF	ADDR match flag. 0: ADDR does not match the received character. 1: ADDR matches the received character, An interrupt is generated if AMIE=1 in the USART_CTL0 register. Set by hardware, when the character defined by ADDR [7:0] is received. Cleared by writing 1 to the AMC in the USART_INTC register.
16	BSY	Busy flag 0: USART reception path is idle.

		1: USART reception path is working.
15:13	Reserved	Must be kept at reset value.
12	EBF	End of block flag. 0: End of Block not reached. 1: End of Block (number of characters) reached. An interrupt is generated if the EBIE=1 in the USART_CTL1 register. Set by hardware when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4. Cleared by writing 1 to EBC bit in USART_INTC register.
11	RTF	Receiver timeout flag. 0: Timeout value not reached. 1: Timeout value reached without any data reception. An interrupt is generated if RTIE bit in the USART_CTL1 register is set. Set by hardware when the RT value, programmed in the USART_RT register has lapsed without any communication. Cleared by writing 1 to RTC bit in USART_INTC register. The timeout corresponds to the CWT or BWT timings in smartcard mode.
10:9	Reserved	Must be kept at reset value.
8	LBDF	LIN break detected flag. 0: LIN Break is not detected. 1: LIN Break is detected. An interrupt will occur if the LBDIE bit is set in USART5_CTL1. Set by hardware when the LIN break is detected. Cleared by writing 1 to LBDC bit in USART_INTC register.
7	TBE	Transmit data register empty. 0: Data is not transferred to the shift register. 1: Data is transferred to the shift register. An interrupt will occur if the TBEIE bit is set in USART_CTL0. Set by hardware when the content of the USART_TDATA register has been transferred into the transmit shift register or writing 1 to TXFCMD bit of the USART_CMD register. Cleared by a write to the USART_TDATA.
6	TC	Transmission completed. 0: Transmission is not completed. 1: Transmission is complete. An interrupt will occur if the TCIE bit is set in USART_CTL0. Set by hardware if the transmission of a frame containing data is completed and if the TBE bit is set. Cleared by writing 1 to TCC bit in USART_INTC register.
5	RBNE	Read data buffer not empty.

		0: Data is not received. 1: Data is received and ready to be read. An interrupt will occur if the RBNEIE bit is set in USART_CTL0. Set by hardware when the content of the receive shift register has been transferred to the USART_RDATA. Cleared by reading the USART_RDATA or writing 1 to RXFCMD bit of the USART_CMD register.
4	IDLEF	IDLE line detected flag. 0: No Idle Line is detected. 1: Idle Line is detected. An interrupt will occur if the IDLEIE bit is set in USART_CTL0. Set by hardware when an Idle Line is detected. It will not be set again until the RBNE bit has been set itself. Cleared by writing 1 to IDLEC bit in USART_INTC register.
3	ORERR	Overrun error 0: No overrun error is detected. 1: Overrun error is detected. An interrupt will occur if the RBNEIE bit is set in USART_CTL0. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2. Set by hardware when the word in the receive shift register is ready to be transferred into the USART_RDATA register while the RBNE bit is set. Cleared by writing 1 to OREC bit in USART_INTC register.
2	NERR	Noise error flag. 0: No noise error is detected. 1: Noise error is detected. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2. Set by hardware when noise error is detected on a received frame. Cleared by writing 1 to NEC bit in USART_INTC register.
1	FERR	Frame error flag. 0: No framing error is detected. 1: Frame error flag or break character is detected. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2. Set by hardware when a de-synchronization, excessive noise or a break character is detected. This bit will be set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame), when USART transmits in smartcard mode. Cleared by writing 1 to FEC bit in USART_INTC register.
0	PERR	Parity error flag. 0: No parity error is detected. 1: Parity error flag is detected. An interrupt will occur if the PERRIE bit is set in USART_CTL0.

Set by hardware when a parity error occurs in receiver mode.

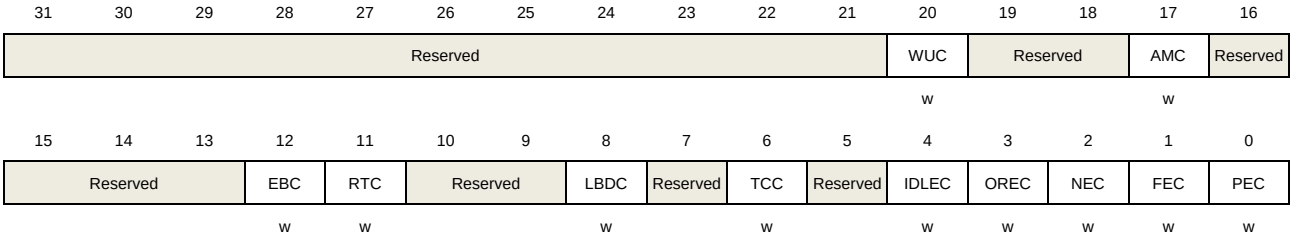
Cleared by writing 1 to PEC bit in USART_INTC register.

Interrupt status clear register (USART_INTC)

Address offset: 0x20

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20	WUC	Wakeup from Deep-sleep mode clear. Writing 1 to this bit clears the WUF bit in the USART_STAT register.
19:18	Reserved	Must be kept at reset value.
17	AMC	ADDR match clear. Writing 1 to this bit clears the AMF bit in the USART_STAT register.
16:13	Reserved	Must be kept at reset value.
12	EBC	End of block clear. Writing 1 to this bit clears the EBF bit in the USART_STAT register.
11	RTC	Receiver timeout clear Writing 1 to this bit clears the RTF flag in the USART_STAT register.
10:9	Reserved	Must be kept at reset value.
8	LBDC	LIN break detected clear. Writing 1 to this bit clears the LBDF flag in the USART_STAT register.
7	Reserved	Must be kept at reset value.
6	TCC	Transmission complete clear. Writing 1 to this bit clears the TC bit in the USART_STAT register.
5	Reserved	Must be kept at reset value.
4	IDLEC	Idle line detected clear. Writing 1 to this bit clears the IDLEF bit in the USART_STAT register.

3	OREC	Overrun error clear. Writing 1 to this bit clears the ORERR bit in the USART_STAT register.
2	NEC	Noise detected clear. Writing 1 to this bit clears the NERR bit in the USART_STAT register.
1	FEC	Frame error flag clear. Writing 1 to this bit clears the FERR bit in the USART_STAT register.
0	PEC	Parity error clear. Writing 1 to this bit clears the PERR bit in the USART_STAT register.

Receive data register (USART_RDATA)

Address offset: 0x24
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							RDATA[8:0]								
							r								

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8:0	RDATA[8:0]	Receive data value The received data character is contained in these bits. The value read in the MSB (bit 7 or bit 8 depending on the data length) will be the received parity bit, if receiving with the parity is enabled (PCEN bit set to 1 in the USART_CTL0 register).

Transmit data register (USART_TDATA)

Address offset: 0x28
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							TDATA[8:0]								
							rw								

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8:0	TDATA[8:0]	Transmit Data value. The transmit data character is contained in these bits. The value written in the MSB (bit 7 or bit 8 depending on the data length) will be replaced by the parity, when transmitting with the parity is enabled (PCEN bit set to 1 in the USART_CTL0 register). This register must be written only when TBE bit in USART_STAT register is set.

USART coherence control register (USART_CHC)

Address offset: 0xC0
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							EPERR	Reserved							
							rc_w0								

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	EPERR	Early parity error flag. This flag will be set as soon as the parity bit has been detected, which is before RBNE flag. This flag is cleared by writing 0. 0: No parity error is detected. 1: Parity error is detected.
7:0	Reserved	Must be kept at reset value.

USART receive FIFO control and status register (USART_RFCS)

Address offset: 0xD0
Reset value: 0x0000 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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFFINT	RFCNT[2:0]			RFF	RFE	RFFIE	RFEN	Reserved							ELNACK
r_w0	r			r	r	rw	rw								rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	RFFINT	Receive FIFO full interrupt flag.
14:12	RFCNT[2:0]	Receive FIFO counter number.
11	RFF	Receive FIFO full flag. 0: Receive FIFO not full. 1: Receive FIFO full.
10	RFE	Receive FIFO empty flag 0: Receive FIFO not empty. 1: Receive FIFO empty.
9	RFFIE	Receive FIFO full interrupt enable. 0: Receive FIFO full interrupt disable. 1: Receive FIFO full interrupt enable.
8	RFEN	Receive FIFO enable. This bit can be set when UESM = 1. 0: Receive FIFO disable. 1: Receive FIFO enable.
7:1	Reserved	Must be kept at reset value.
0	ELNACK	Early NACK when smartcard mode is selected. The NACK pulse occurs 1/16 bit time earlier when the parity error is detected. 0:Early NACK disable when smartcard mode is selected. 1:Early NACK enable when smartcard mode is selected.

21. Inter-integrated circuit interface (I2C)

21.1. Inter-integrated circuit interface (I2Cx, x=0, 1)

21.1.1. Overview

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. I2C bus uses two serial lines: a serial data line, SDA, and a serial clock line, SCL.

The I2C interface implements standard I2C protocol with standard mode, fast mode and fast mode plus as well as CRC calculation and checking, SMBus (system management bus), PMBus (power management bus) and SAM_V (secure access and control module for validation) mode. It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

21.1.2. 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), fast mode (up to 400 kHz) and fast mode plus (up to 1MHz).
- 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.
- Supports SAM_V mode.

21.1.3. Function overview

Figure 21-1. I2C module block diagram below provides details of the internal configuration of the I2C interface.

Figure 21-1. I2C module block diagram

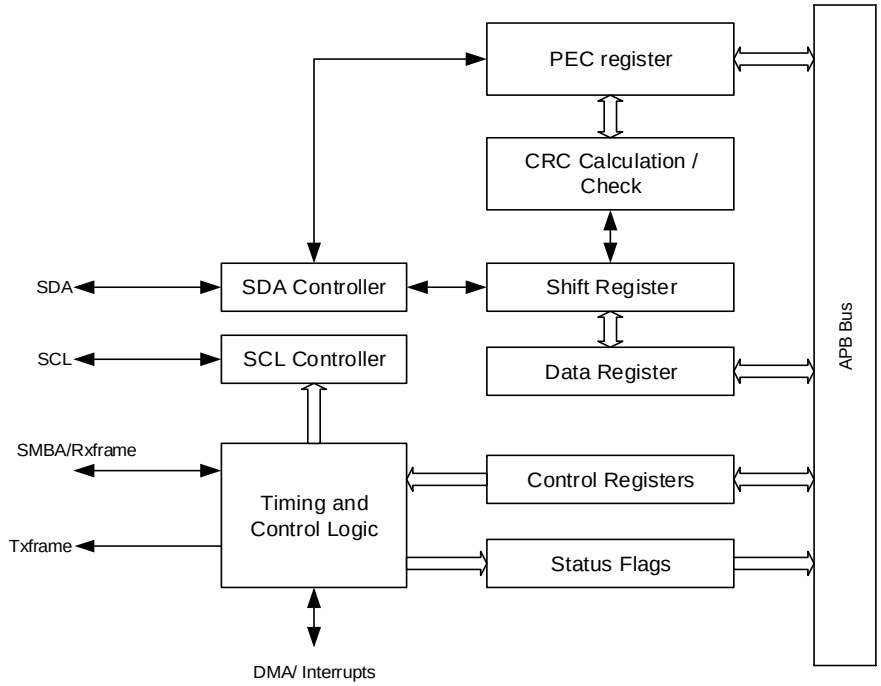


Table 21-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors)

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

SDA and SCL lines

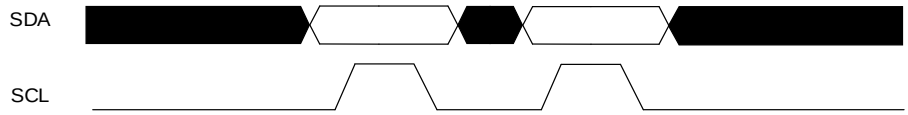
The I2C module has two external lines, the serial data SDA and serial clock SCL lines. The two wires carry information between the devices connected to the bus. Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via current-source or pull-up resistor. When the bus is free, 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, up to 400 Kbit/s in the fast mode and up to 1Mbit/s in the fast mode plus if the FMPEN bit in I2C_CTL2 is set. 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}.

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 SDA line can only change when the clock signal on the SCL line is LOW (see ***Figure 21-2. Data validation***). One clock pulse is generated for each data bit to be transferred.

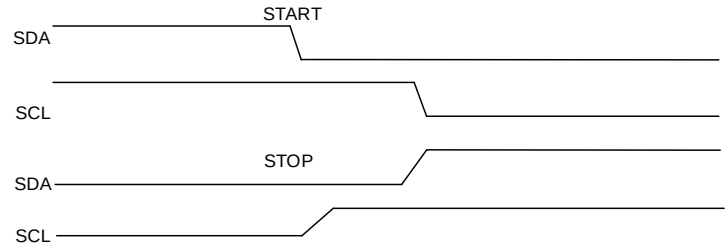
Figure 21-2. Data validation



START and STOP signal

All transmissions begin with a START and are terminated by a STOP (see ***Figure 21-3. START and STOP signal***). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START signal. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP signal.

Figure 21-3. START and STOP signal



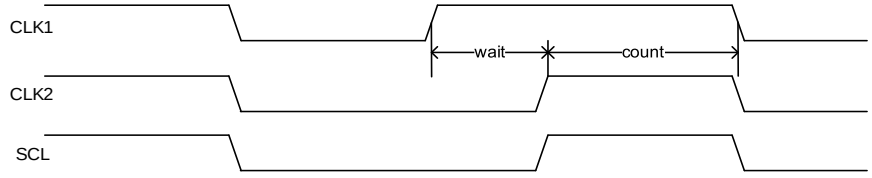
Clock synchronization

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 completes 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 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 21-4. 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 period enter

a HIGH wait-state during this time.

Figure 21-4. Clock synchronization



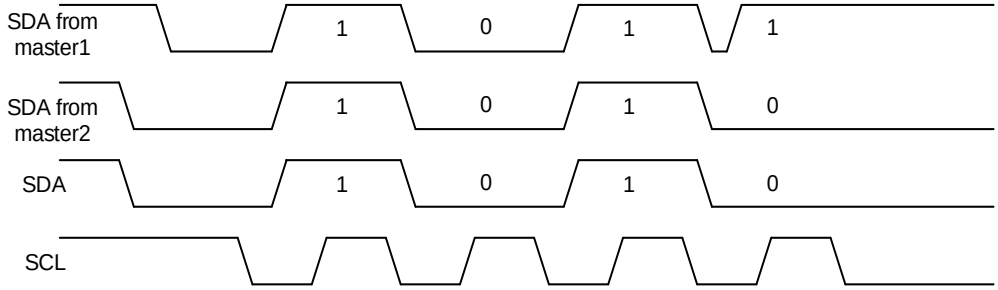
Arbitration

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 signal within the minimum hold time of the START signal which results in a valid START signal 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 whether the SDA level matches what it has been sent. This process may take many bits. Two masters can even complete an entire transmission 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 transmission.

Figure 21-5. SDA line arbitration



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 be operated as either a transmitter or receiver, depending on the function of the device.

An I2C slave will continue to detect addresses after a START signal on I2C bus and compare the detected address with its slave address which is programmed by software. Once the two addresses match with each other, the I2C slave will send an ACK to the I2C bus and respond 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 responds to a General Call Address (0x00). The I2C block supports both 7-bit and 10-bit address modes.

An I2C master always initiates or ends a transfer using START or STOP signal and it's also responsible for SCL clock generation.

Figure 21-6. I2C communication flow with 7-bit address

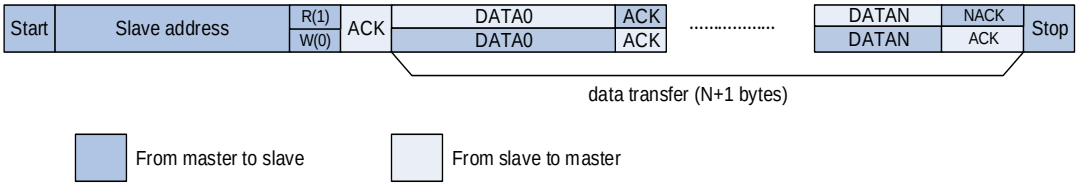


Figure 21-7. I2C communication flow with 10-bit address (Master Transmit)

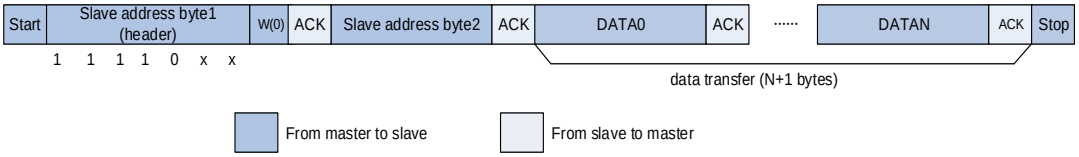
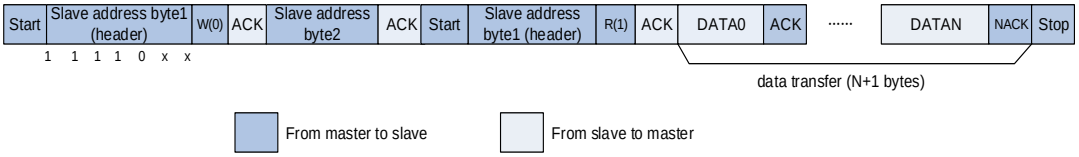


Figure 21-8. I2C communication flow with 10-bit address (Master Receive)



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 as 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. After sending a START signal on I2C bus, it changes into master mode. The I2C changes back to slave mode after sending a STOP signal on I2C bus.

- **Programming model in slave transmitting mode**

As is shown in ***Figure 21-9. Programming model for slave transmitting (10-bit address mode)***, the following software procedure should be followed if users wish to transmit data in slave transmitter mode:

1. First of all, 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 signal followed by address on I2C bus.

2. After receiving a START signal 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 signal followed by a header to the I2C bus. The slave sets ADDSEND bit again after it detects the repeated START signal and the following header. The ADDSEND bit must be cleared by software 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 byte written 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.

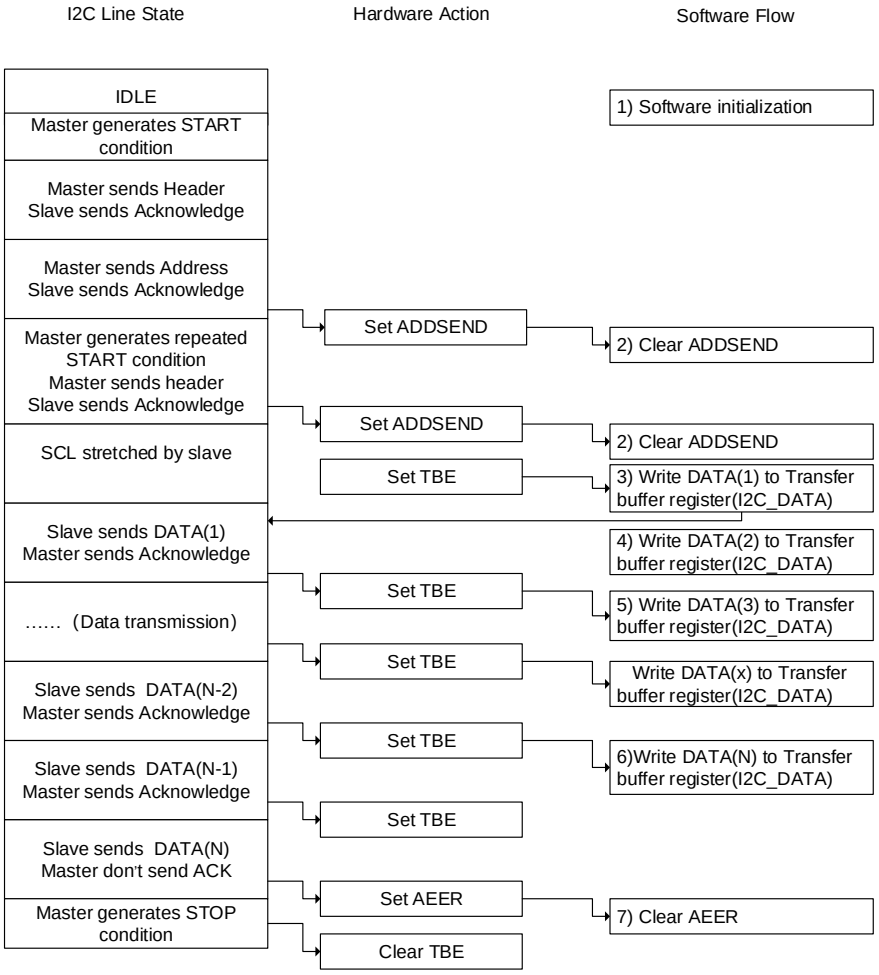
4. During the transmission of the first byte, software can write the second byte to I2C_DATA, and this time TBE is cleared because neither I2C_DATA nor shift register is empty.

5. After the transmission of the first byte, the TBE bit will be set, the software can write the third byte to the I2C_DATA register and TBE is cleared. After this, any time TBE is set, software can write a byte to I2C_DATA as long as there is still data to be transmitted.

6. During the transmission of the second last byte, software writes 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 signal.

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 signal on I2C bus and sets AERR (Acknowledge Error) bit to notify software that the transmission completes. Software clears AERR bit by writing 0 to it.

Figure 21-9. Programming model for slave transmitting (10-bit address mode)



- **Programming model in slave receiving mode**

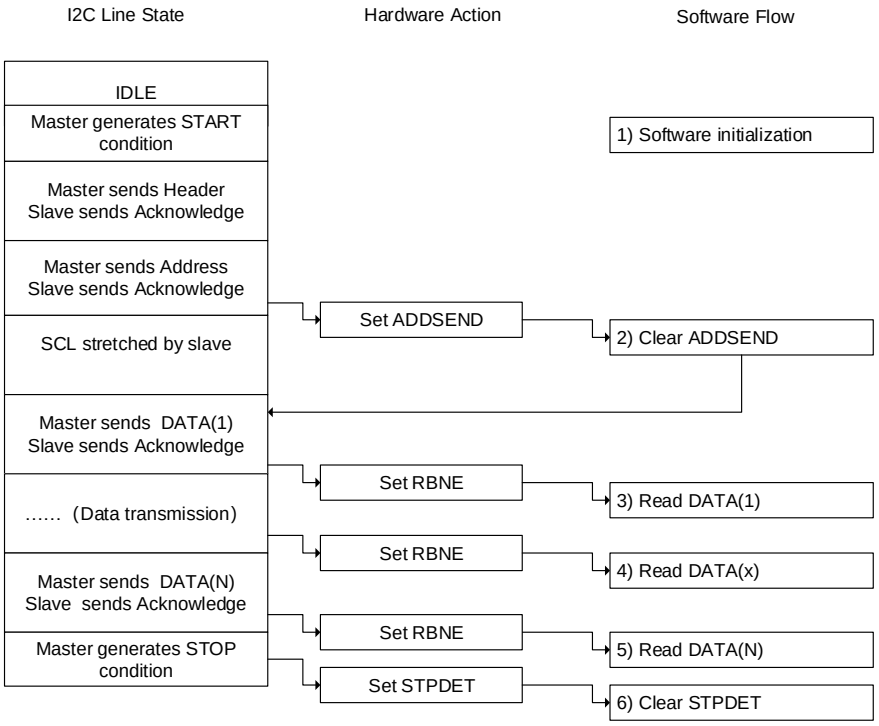
As is shown in ***Figure 21-10. Programming model for slave receiving (10-bit address mode)***, the following software procedure should be followed if users wish to receive data in slave receiver mode:

1. First of all, 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 signal followed by address on I2C bus.
2. After receiving a START signal followed by a matched 7-bit or 10-bit address, the I2C hardware sets the ADDSEND bit in I2C status register 0, 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 on 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 the last byte is received, RBNE is set. Software reads the last byte.

6. STPDET bit is set when I2C detects a STOP signal on I2C bus and software reads I2C_STAT0 and then writes I2C_CTL0 to clear the STPDET bit.

Figure 21-10. Programming model for slave receiving (10-bit address mode)



- **Programming model in master transmitting mode**

As is shown in ***Figure 21-11. Programming model for master transmitting (10-bit address mode)***, the following software procedure should be followed if users wish to make transaction in master transmitter mode:

1. First of all, 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 signal followed by address on I2C bus.

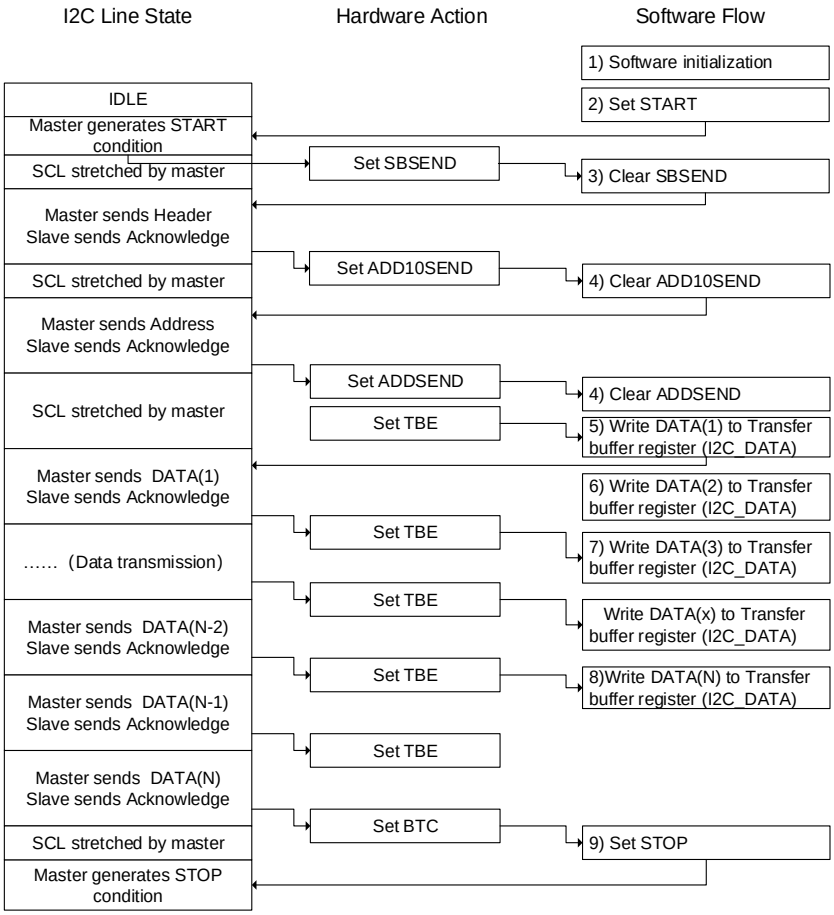
2. Software sets START bit requesting I2C to generate a START signal on I2C bus.

3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C_STAT0 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 which has been sent is header of a 10-bit address, the hardware sets ADD10SEND bit after sending the 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 has been 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 writes the first byte data to I2C_DATA register, but the TBE will not be cleared because the byte written in I2C_DATA is moved to internal shift register immediately. The I2C begins to transmit data to I2C bus as soon as the shift register is not empty.
6. During the transmission of the first byte, 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 is still data to be transmitted.
8. During the transmission of the second last byte, software writes the last data to I2C_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be asserted after the transmission of the byte and not be cleared until a STOP signal.
9. After sending the last byte, I2C master sets BTC bit because both the shift register and I2C_DATA are empty. Software should set the STOP bit to generate a STOP signal, then the I2C clears both TBE and BTC flags.

Figure 21-11. Programming model for master transmitting (10-bit address mode)



- **Programming model in master receiving mode**

In master receiving mode, a master is responsible for generating NACK for the last byte

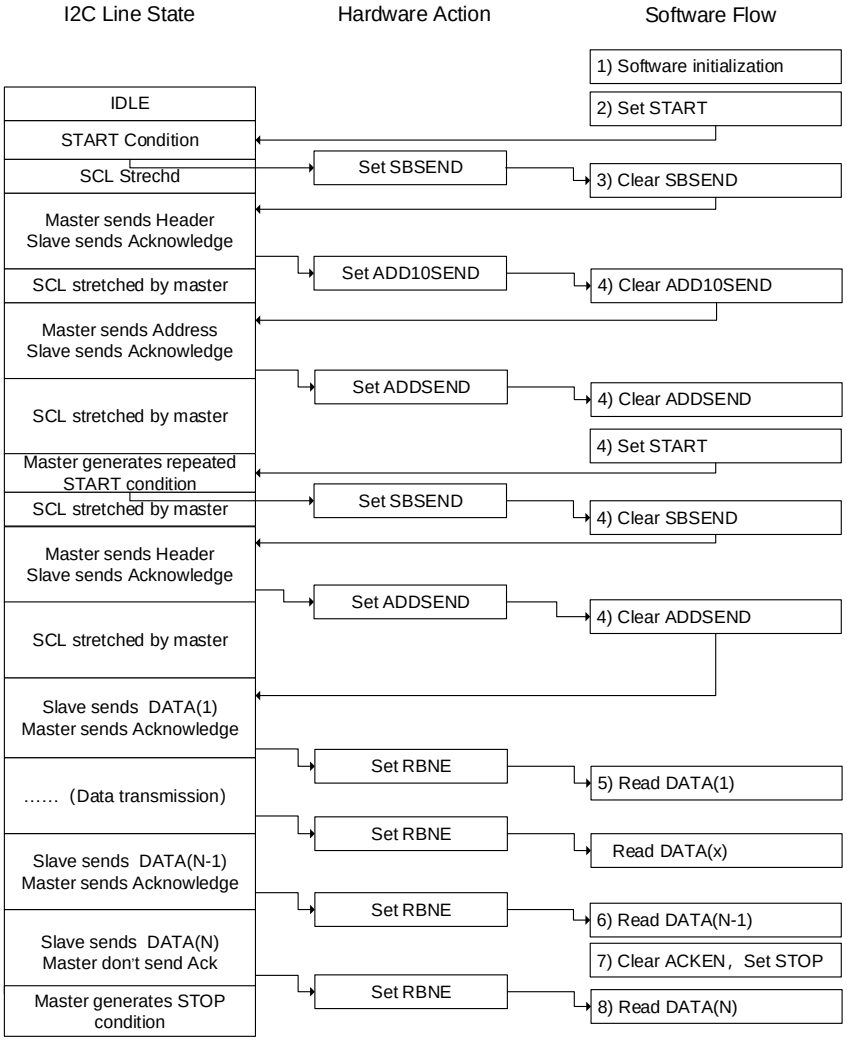
reception and then sending a STOP signal on I2C bus. So, special attention should be paid to ensure the correct ending of data reception. Two solutions for master receiving are provided here for applications: Solution A and B. Solution A requires the software's quick response to I2C events, while Solution B doesn't.

Solution A

1. First of all, 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 signal followed by address on I2C bus.
2. Software sets START bit requesting I2C to generate a START signal on I2C bus.
3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C_STAT0 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 which has been sent is header of a 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 has been 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 signal 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 (N-1) 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 will be sent for the last byte.
8. After the last byte is received, RBNE is set. Software reads the last byte. Since ACKEN has been cleared in the previous step, I2C doesn't send ACK for the last byte and it generates a STOP signal after the transmission of the last byte.

The 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 21-12. Programming model for master receiving using Solution A (10-bit address mode)



Solution B

1. First of all, 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 signal followed by address on I2C bus.

2. Software sets START bit requesting I2C to generate a START signal on I2C bus.

3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C_STAT0 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 which has been 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 has been 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 signal 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 until the master receives N-3 bytes.

As shown in ***Figure 21-13. Programming model for master receiving mode using solution B (10-bit address mode)***, 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. Master doesn't send an ACK for the last byte because ACKEN is already cleared.

8. After the last byte is received, both BTC and RBNE are set again, and SCL is stretched low. Software sets STOP bit and master sends out a STOP signal 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.

The above steps require that byte number N>2. N=1 and N=2 are similar:

N=1

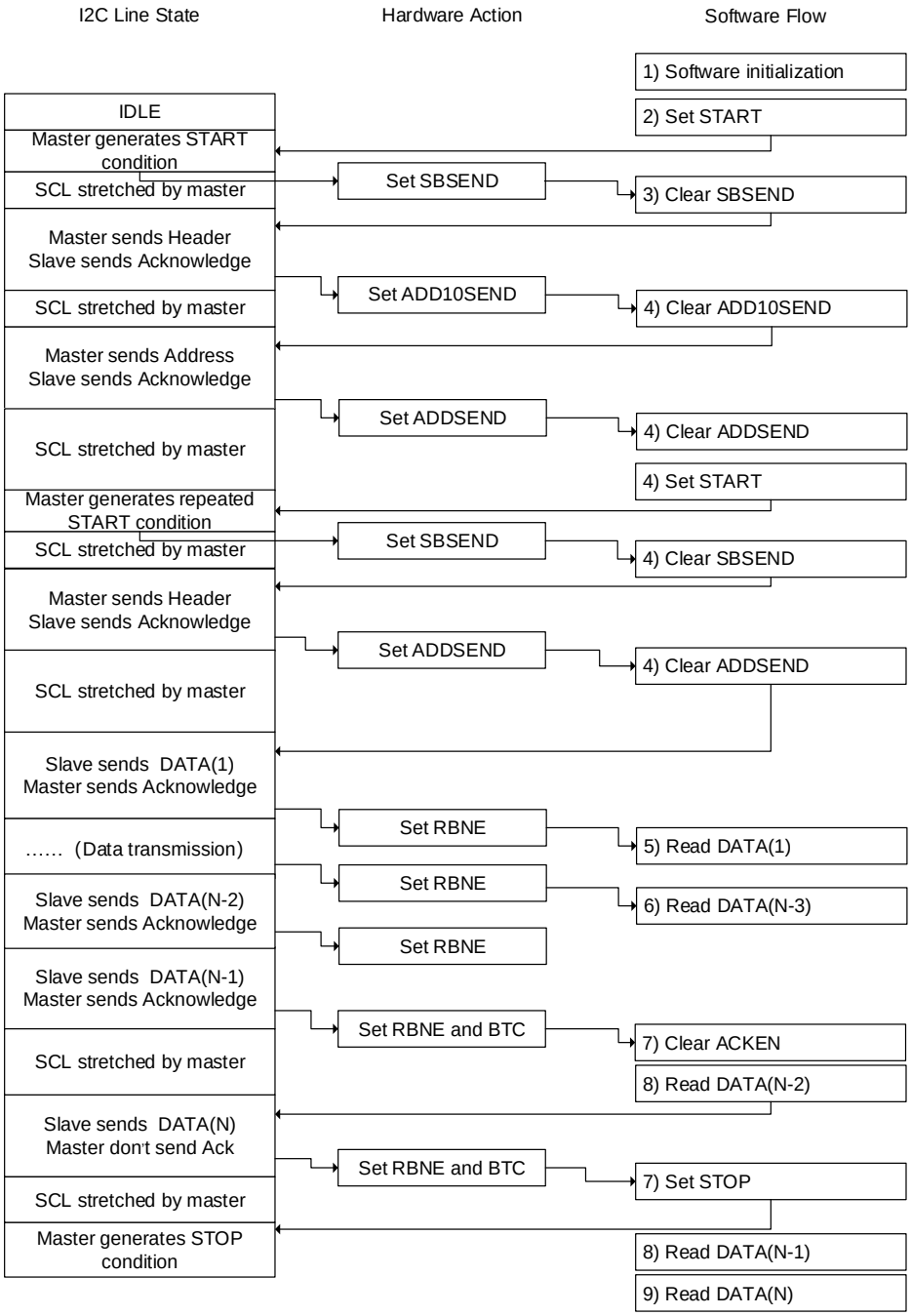
In Step4, software should reset ACKEN bit before clearing ADDSEND bit and set STOP bit after clearing ADDSEND bit. Step 5 is the last step when N=1.

N=2

In Step 2, software should set POAP bit before setting 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 read I2C_DATA twice.

Figure 21-13. Programming model for master receiving mode using solution B (10-bit

address mode)



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 bits are set in transmitting mode, the transmitter stretches the SCL line low until the transfer buffer register is filled with the next data to be transmitted. When the RBNE and BTC bits are set in receiving mode, 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

SS 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.

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 to be high overloaded. The DMA controller can be used to process TBE and RBNE flags: each time TBE or RBNE is asserted, DMA controller does a read or write operation automatically. It reduces the load on the CPU. See the DMA section for details on how to configure DMA.

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 the I2C transfer. When the configured number of bytes have been transferred, the DMA controller generates End of Transfer (EOT) interrupt. DMA will send an End of Transmission (EOT) signal to the I2C interface and generates a DMA full transfer finish interrupt.

When a master receives two or more bytes, the DMALST bit in the I2C_CTL1 register should be set. The I2C master will send NACK after the last byte. The STOP bit can be set by software to generate a STOP signal in the ISR of the DMA full transfer finish 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 signal after clearing the ADDSEND status, or in the ISR of the DMA full transfer finish interrupt.

Packet error checking

There is a CRC-8 calculator in I2C block to perform PEC (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 and PECTRANS bit are set.

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 transmission 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 is realized based on I2C hardware and it uses I2C hardware addressing, but it adds the 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 allows 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 this protocol there is a very useful distinction between a system host and all the other devices in the system, that is the host provides address assignment function.

- **Time-out feature**

  SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency is 10 kHz to prevent locking up the bus. I2C can be a 'DC' bus, which means that a slave device stretches the master clock when performing some routines while the master is accessing it. This will notify the master that the slave is busy but does not want to lose the communication. The slave device will continue the communication after its task is completed. There is no limit in the I2C bus protocol of 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 solve the problem. 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 byte is appended at the end of each transaction. The byte is a CRC-8 checksum of 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 which is based on the I2C multi-master mode but it can pass more data.

- **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, respond 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 values.
2. In order to support address resolution protocol (ARP) (ARPEN=1), the software should respond to HSTSMB flag in SMBus Host Mode (SMBSEL =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 respond to SMBALT flag and implement the related function.

SAM_V support

To support the SAM_V standard, two additional pins are added to the I2C module: txframe and rxframe. Txframe is an output pin, in master mode, it indicates the I2C is busy when it is asserted. Rxframe is an input pin that is supposed to be multiplexed together with the SMBALERT signal.

The SAM_V mode is enabled by setting the SAMEN bit of the I2C_SAMCS register. The status of the txframe and rxframe pin can be reflected by the RFR, RFF, TFR, TFF, RXF, and TXF flags of the I2C_SAMCS register. I2C interrupts will be generated if the corresponding interrupt enable bits are set.

Status, errors and interrupts

There are several status and error flags in I2C, and interrupts may be asserted from these flags by setting some register bits (refer to ***Register definition*** for detail).

Table 21-2. Event status flags

Event Flag Name	Description
SBSEND	START signal sent (master)
ADDSEND	Address sent or received
ADD10SEND	Header of 10-bit address sent
STPDET	STOP signal detected
BTC	Byte transmission completed
TBE	I2C_DATA is empty when transmitting
RBNE	I2C_DATA is not empty when receiving
RFR	SAM_V mode rxframe pin rising edge is detected
RFF	SAM_V mode rxframe pin falling edge is detected
TFR	SAM_V mode txframe pin rising edge is detected
TFF	SAM_V mode txframe pin falling edge is detected

Table 21-3. Error flags

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
SMBTO	Bus timeout in SMBus mode
SMBALT	SMBus Alert

21.1.4. Register definition

I2C0 base address: 0x4000 5400

I2C1 base address: 0x4000 5800

Control register 0 (I2C_CTL0)

Address offset: 0x00

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
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
31:16	Reserved	Must be kept at reset value.
15	SRESET	Software resets 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 at 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 sets and clears this bit while hardware clears this bit when PEC is transferred or START / STOP signal is detected or I2CEN=0. 0: Don’t transfer PEC value 1: Transfer PEC value
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 signal on I2C bus This bit is set and cleared by software and set by hardware when SMBus timeout and cleared by hardware when STOP signal is detected. 0: STOP will not be sent 1: STOP will be sent
8	START	Generate a START signal on I2C bus This bit is set and cleared by software and cleared by hardware when a START signal is 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 respond to a General Call 1: Slave will respond to a General Call
5	PECEN	PEC calculation enable 0: PEC calculation disable 1: PEC calculation enable
4	ARPEN	ARP protocol enable in SMBus mode 0: ARP is disabled 1: ARP is enabled
3	SMBSEL	SMBus type selection 0: Device 1: Host
2	Reserved	Must be kept at 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

Control register 1 (I2C_CTL1)

Address offset: 0x04

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			DMALST	DMAON	BUFIE	EVIE	ERRIE	Reserved	I2CCLK[6:0]						
			rw	rw	rw	rw	rw		rw						

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12	DMALST	DMA last transfer configure 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 switched off 1: DMA mode switched on
10	BUFIE	Buffer interrupt enable 0: Buffer interrupt is disabled, TBE = 1 or RBNE = 1 when EVIE=1 will not generate an interrupt. 1: Buffer interrupt is enabled, which means that interrupt will be generated when TBE = 1 or RBNE = 1 if EVIE=1.
9	EVIE	Event interrupt enable 0: Event interrupt is disabled 1: Event interrupt is enabled, which 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 is disabled 1: Error interrupt is enabled, which means that interrupt will be generated when BERR, LOSTARB, AERR, OUERR, PECERR, SMBTO or SMBALT flag is asserted.
7	Reserved	Must be kept at reset value.
6:0	I2CCLK[6:0]	I2C peripheral clock frequency I2CCLK[6:0] should be the frequency of input APB1 clock in MHz which is at least 2.

0000000 - 0000001: Not allowed

0000010 - 1011010: 2MHz~90MHz

1011011 - 1111111: 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. In I2C fast mode plus, the frequencies of APB1 must be equal or greater than 24MHz.

Slave address register 0 (I2C_SADDR0)

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
ADDFORMAT	Reserved					ADDRESS[9:8]		ADDRESS[7:1]							ADDRESS0
rw						rw		rw							rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	ADDFORMAT	Address format for the I2C slave 0: 7-bit address 1: 10-bit address
14:10	Reserved	Must be kept at 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

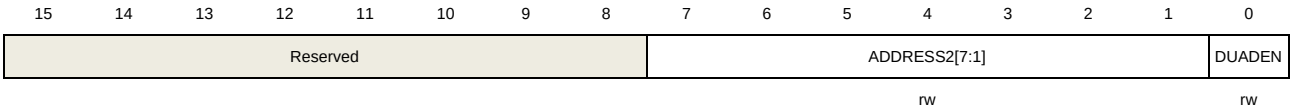
Slave address register 1 (I2C_SADDR1)

Address offset: 0x0C

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															



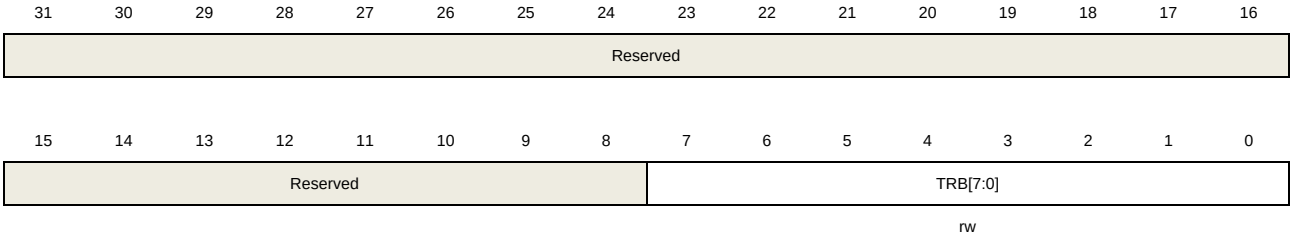
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:1	ADDRESS2[7:1]	The second I2C address for the slave in Dual-Address mode
0	DUADEN	Dual-Address mode enable 0: Dual-Address mode is disabled 1: Dual-Address mode is enabled

Transfer buffer register (I2C_DATA)

Address offset: 0x10

Reset value: 0x0000 0000

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



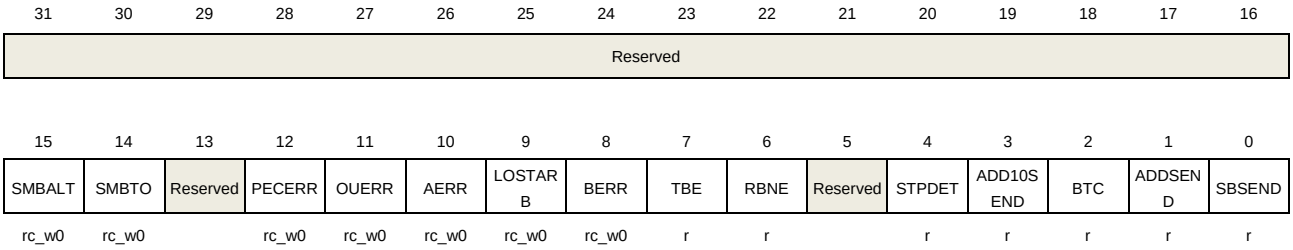
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	TRB[7:0]	Transmission or reception data buffer

Transfer status register 0 (I2C_STAT0)

Address offset: 0x14

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions

31:16	Reserved	Must be kept at reset value.
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 and Alert address received (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 be kept at reset value.
12	PECERR	PEC error when receiving data This bit is set by hardware and cleared by writing 0. 0: Received PEC matches the calculated PEC 1: Received PEC doesn't match the calculated PEC, 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	Bus error A bus error occurs when an unexpected START or STOP signal 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 byte in shift register will be moved to I2C_DATA immediately. 0: I2C_DATA is empty 1: I2C_DATA is not empty, software can read
5	Reserved	Must be kept at reset value.
4	STPDET	STOP signal is detected in slave mode This bit is set by hardware and cleared by reading I2C_STAT0 and then writing I2C_CTL0. 0: STOP signal not detected in slave mode 1: STOP signal 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 is sent in master mode 1: Header of 10-bit address is sent in master mode
2	BTC	Byte transmission is 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 if SCL stretching enabled. This bit is set by hardware and cleared by 3 ways as follow: 1. Software clearing: reading I2C_STAT0 followed by reading or writing I2C_DATA 2. Hardware clearing: sending the STOP signal or START signal 3. Bit 0 (I2CEN bit) of the I2C_CTL0 is reset. 0: BTC not asserted 1: BTC asserted
1	ADDSEND	Address is sent and ACK is received in master mode or address is received and matches with its own address in slave mode. This bit is set by hardware and cleared by reading I2C_STAT0 and reading I2C_STAT1. 0: In slave mode, no address is received or the received address does not match witih its own address. In master mode, no address is sent or address has been sent but not received the ACK from slave. 1: In slave mode, address is received and matches witih its own address. In master

		mode, address has been sent and receives the ACK from slave.
0	SBSEND	START signal is sent out in master mode This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA. 0: No START signal sent 1: START signal sent

Transfer status register 1 (I2C_STAT1)

Address offset: 0x18

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
PECV[7:0]								DUMODF	HSTSMB	DEFSMB	RXGC	Reserved	TR	I2CBSY	MASTER
r								r	r	r	r		r	r	r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:8	PECV[7:0]	Packet Error Checking value that calculated by hardware when PEC is enabled.
7	DUMODF	Dual flag in slave mode indicates which address matches with the address in Dual-Address mode This bit is cleared by hardware after a STOP or a START signal or I2CEN=0 0: The address matches with SADDR0 address 1: The address matches with SADDR1 address
6	HSTSMB	SMBus host header detected in slave mode This bit is cleared by hardware after a STOP or a START signal or I2CEN=0 0: No SMBus host header is detected 1: SMBus host header is detected
5	DEFSMB	Default address of SMBus device This bit is cleared by hardware after a STOP or a START signal or I2CEN=0. 0: The default address has not been received for SMBus device 1: The default address has been received for SMBus device
4	RXGC	General call address (0x00) received. This bit is cleared by hardware after a STOP or a START signal or I2CEN=0. 0: No general call address received 1: General call address received

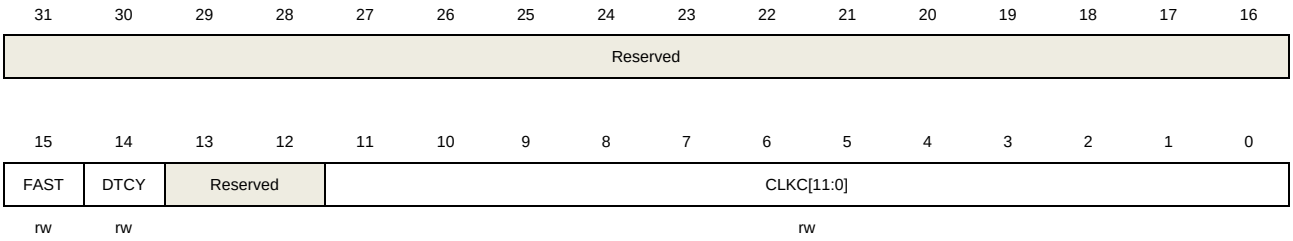
3	Reserved	Must be kept at reset value.
2	TR	Transmitter or receiver This bit indicates whether the I2C is a transmitter or a receiver. It is cleared by hardware after a STOP or a START signal or I2CEN=0 or LOSTARB=1. 0: Receiver 1: Transmitter
1	I2CBSY	Busy flag This bit is cleared by hardware after a STOP signal 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 set by hardware when a START signal generates. This bit is cleared by hardware after a STOP signal or I2CEN=0 or LOSTARB=1. 0: Slave mode 1: Master mode

Clock configure register (I2C_CKCFG)

Address offset: 0x1C

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	FAST	I2C speed selection in master mode 0: Standard speed 1: Fast speed
14	DTCY	Duty cycle in fast mode or fast mode plus 0: $T_{low}/T_{high}=2$ 1: $T_{low}/T_{high}=16/9$
13:12	Reserved	Must be kept at 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 or fast mode plus, if DTCY=0:

T_{high}=CLKC*T_{PCLK1}，T_{low}=2*CLKC*T_{PCLK1}

In fast speed mode or fast mode plus, if DTCY=1:

T_{high}=9*CLKC*T_{PCLK1}，T_{low}=16*CLKC*T_{PCLK1}

Note: If DTCY is 0, when PCLK1 is an integral multiple of 3, the baud rate will be more accurate. If DTCY is 1, when PCLK1 is an integral multiple of 25, the baud rate will be more accurate.

Rise time register (I2C_RT)

Address offset: 0x20

Reset value: 0x0000 0002

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									RISETIME[6:0]						
									rw						

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6:0	RISETIME[6:0]	Maximum rise time in master mode The RISETIME value should be the maximum SCL rise time incremented by 1.

SAM control and status register (I2C_SAMCS)

Address offset: 0x80

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
RFR	RFF	TFR	TFF	Reserved		RXF	TXF	RFRIE	RFFIE	TFRIE	TFFIE	Reserved		STOEN	SAMEN
rc_w0	rc_w0	rc_w0	rc_w0			r	r	rw	rw	rw	rw			rw	rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	RFR	Rxframe rise flag, cleared by software by writing 0
14	RFF	Rxframe fall flag, cleared by software by writing 0

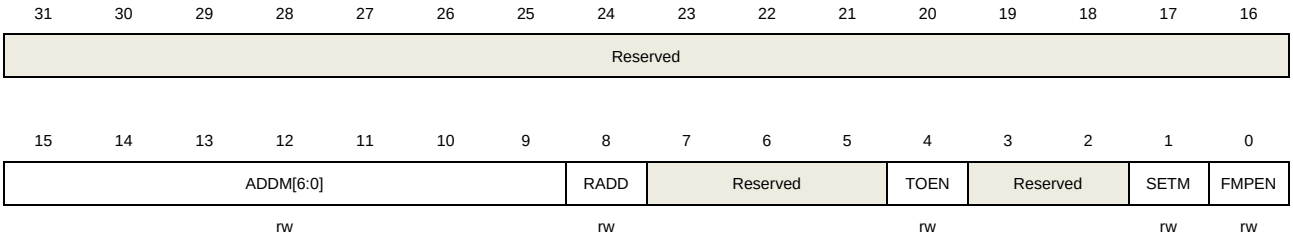
13	TFR	Txframe rise flag, cleared by software by writing 0
12	TFF	Txframe fall flag, cleared by software by writing 0
11:10	Reserved	Must be kept at reset value.
9	RXF	Level of rxframe signal
8	TXF	Level of txframe signal
7	RFRIE	Rxframe rise interrupt enable 0: Rxframe rise interrupt disabled 1: Rxframe rise interrupt enabled
6	RFFIE	Rxframe fall interrupt enable 0: Rxframe fall interrupt disabled 1: Rxframe fall interrupt enabled
5	TFRIE	Txframe rise interrupt enable 0: Txframe rise interrupt disabled 1: Txframe rise interrupt enabled
4	TFFIE	Txframe fall interrupt enable 0: Txframe fall interrupt disabled 1: Txframe fall interrupt enabled
3:2	Reserved	Must be kept at reset value.
1	STOEN	SAM_V interface timeout detect enable 0: SAM_V interface timeout detect disabled 1: SAM_V interface timeout detect enabled
0	SAMEN	SAM_V interface enable 0: SAM_V interface disabled 1: SAM_V interface enabled

Control register 2 (I2C_CTL2)

Address offset: 0x90

Reset value: 0x0000 FE00

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



Bits	Fields	Descriptions

31:16	Reserved	Must be kept at reset value.
15:9	ADDM[6:0]	Defines which bits of register ADDRESS[7:1] are compared with an incoming address byte, and which bits are ignored. Any bit set to 1 in ADDM[6:0] enables comparisons with the corresponding bit in ADDRESS[7:1]. Bits set to 0 are ignored (can be either 0 or 1 in the incoming address).
8	RADD	The received 7 bit slave address or the [7:0] bits of the received 10 bit slave address will be recorded in the transfer buffer register (I2C_DATA) when this bit is set.
7:5	Reserved	Must be kept at reset value.
4	TOEN	Timeout calculation enable. This bit is used to enable the 25ms SCL low state timeout counter when the SMBA mode is not enabled. 0: disable the SCL low state timeout counter 1: enable the SCL low state timeout counter
3:2	Reserved	Must be kept at reset value.
1	SETM	Start Early Termination Mode. This bit describe how to deal with the low state of the SCL line when I2C is in the SCL high state following the start phase. 0: terminate the SCL high state and start to count the SCL low state. (as the standard I2C protocol) 1: do the same thing as arbitration lost
0	FMPEN	Fast mode plus enable. The I2C device supports up to 1MHz when this bit is set.

Control and status register (I2C_CS)

Address offset: 0x94

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						STPSENDIE	STLOIE	Reserved						STPSEND	STLO
						rw	rw							rc_w0	r_w0

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	STPSENDIE	Interrupt enable for stop condition sent 0: interrupt disable 1: interrupt enable

8	STLOIE	Interrupt enable for start lost 0: interrupt disable 1: interrupt enable
7:2	Reserved	Must be kept at reset value.
1	STPSEND	Stop condition sent out in master mode This bit is set by hardware and cleared by software write 0 0: No STOP condition sent 1: STOP condition sent
0	STLO	Start lost flag. This bit is set when low level of the SCL line occurs when I2C is in the start hold timing phase ($t_{HD: STA}$) if the SETM bit is set to 1. This bit is set by hardware and cleared by software write 0 0: start lost not occur 1: start lost occurred

Status clear register (I2C_STATC)

Address offset: 0x98

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
SRCEN	Reserved										STOPFC	ADD10SENDC	BTCC	ADDSENDC	SBSENDC
rw											w	w	w	w	w

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	SRCEN	Status register clear enable. When this bit is set, the STOPF, ADD10SEND, BTC, ADDSEND, SBSEND status bits can be cleared by writing 1 to the STOPFC, ADD10SENDC, BTCC, ADDSENDC and SBSENDC registers respectively. Once this bit is set the STOPF, ADD10SEND, BTC, ADDSEND, SBSEND status bits can not be cleared by other software sequences. 0: disable the clear function 1: enable the clear function
14:5	Reserved	Must be kept at reset value.
4	STOPFC	STOPF status clear When SRCEN bit is set to 1, software can clear the STPDET bit of I2C_STAT0 by writing 1 to this bit

3	ADD10SENDC	ADD10SEND status clear When SRCEN bit is set to 1, software can clear the ADD10SEND bit of I2C_STAT0 by writing 1 to this bit
2	BTCC	BTC status clear When SRCEN bit is set to 1, software can clear the BTC bit of I2C_STAT0 by writing 1 to this bit
1	ADDSENDC	ADDSEND status clear When SRCEN bit is set to 1, software can clear the ADDSEND bit of I2C_STAT0 by writing 1 to this bit
0	SBSENDC	Start send status clear. When SRCEN bit is set to 1, software can clear the SBSEND bit of I2C_STAT0 by writing 1 to this bit

21.2. Inter-integrated circuit interface (I2Cx, x=2)

21.2.1. Overview

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. I2C bus uses two serial lines: a serial data line, SDA, and a serial clock line, SCL.

The I2C interface implements standard I2C protocol with standard mode, fast mode and fast mode plus 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.

21.2.2. 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.
- Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask).
- Programmable setup time and hold time.
- Multi-master capability.
- Supports standard mode (up to 100 kHz) and fast mode (up to 400 kHz) and fast mode plus (up to 1MHz).
- Configurable SCL stretching in slave mode.
- Supports DMA mode.
- SMBus 3.0 and PMBus 1.3 compatible.
- Optional PEC (packet error checking) generation and check.
- Programmable analog and digital noise filters.
- Wakeup from Deep-sleep mode, Deep-sleep 1 mode and Deep-sleep 2 mode on I2C address match.
- Independent clock from PCLK.

21.2.3. Function overview

Figure 21-14. I2C module block diagram below provides details on the internal configuration of the I2C interface.

Figure 21-14. I2C module block diagram

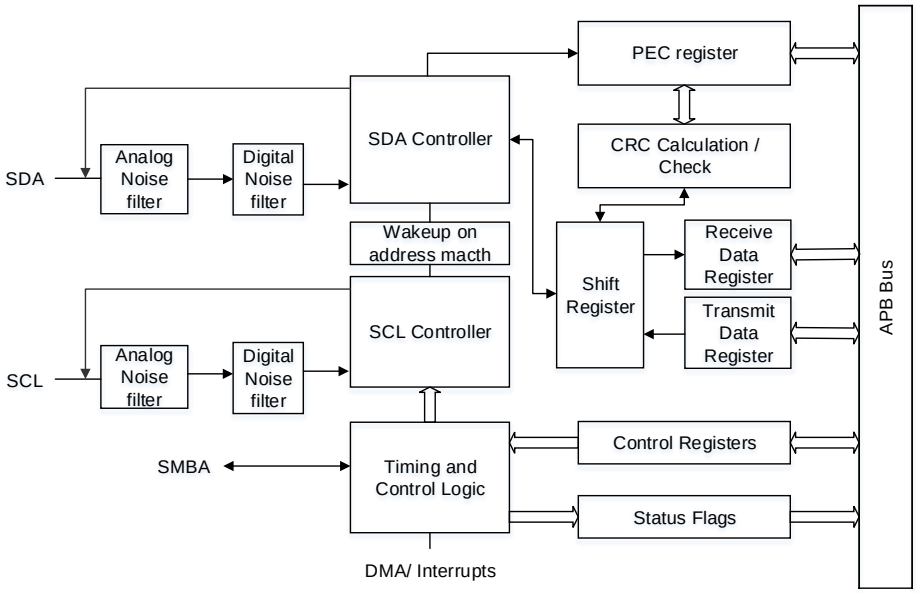


Table 21-4. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors)

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
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

Clock requirements

The I2C clock is independent of the PCLK frequency, so that the I2C can be operated independently.

This I2C clock (I2CCLK) can be selected from the following three clock sources:

- PCLK1: APB1 clock (default value)
- IRC8M: internal 8 MHz RC
- SYSCLK: system clock

The I2CCLK period t_{I2CCLK} must match the conditions as follows:

- $t_{I2CCLK} < (t_{LOW} - t_{filters})/4$
- $t_{I2CCLK} < t_{HIGH}$

with:

t_{LOW}: SCL low time

t_{HIGH}: SCL high time

$t_{filters}$: When the filters are enabled, represent the delays by the analog filter and digital filter.

Analog filter delay is maximum 160ns. Digital filter delay is DNF[3:0]×t_{I2CCLK}.

The period of PCLK clock t_{PCLK} match the conditions as follows:

- t_{PCLK}<4/3*t_{SCL}

with:

t_{SCL}: the period of SCL

Note: When the I2C kernel is provided by PCLK, this clock must match the conditions for t_{I2CCLK}.

I2C communication flow

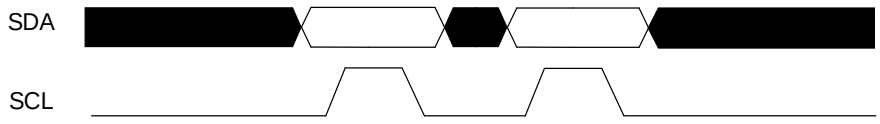
An I2C device is able to transmit or receive data whether it's a master or a slave, thus, there're 4 operation modes for an I2C device:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

- **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 21-15. Data validation***). One clock pulse is generated for each data bit transferred.

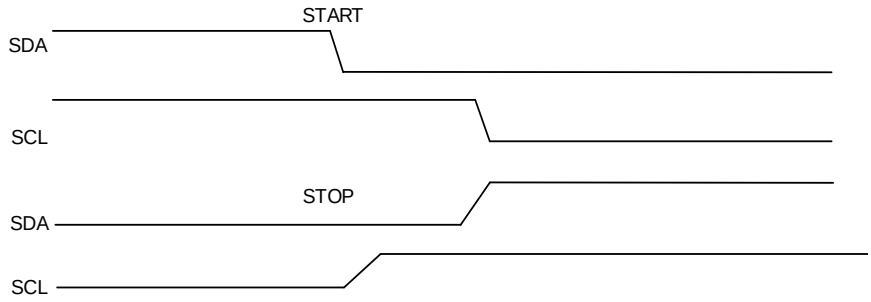
Figure 21-15. Data validation



- **START and STOP signal**

All transactions begin with a START and are terminated by a STOP (see ***Figure 21-16. START and STOP signal***). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START signal. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP signal.

Figure 21-16. START and STOP signal



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. It operates in slave mode by default. When it generates a START signal, the interface automatically switches from slave to master. If an arbitration loss or a STOP generation occurs, then the interface switches from master to slave, allowing multimaster capability.

An I2C slave will continue to detect addresses after a START signalon 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.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START signalcontain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in master mode.

A 9th clock pulse follows the 8 clock cycles of byte transmission, during which the receiver must send an acknowledge bit to the transmitter. Acknowledge can be enabled or disabled by software.

An I2C master always initiates or end a transfer using START or STOP signaland it's also responsible for SCL clock generation.

In master mode, if AUTOEND=1, the STOP signalis generated automatically by hardware. If AUTOEND=0, the STOP signalgenerated by software, or the master can generate a RESTART signalto start a new transfer.

Figure 21-17. I2C communication flow with 10-bit address (Master Transmit)

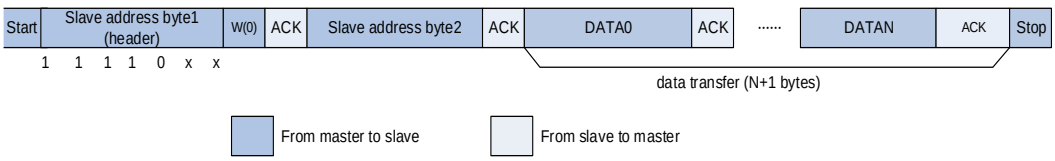


Figure 21-18. I2C communication flow with 7-bit address (Master Transmit)

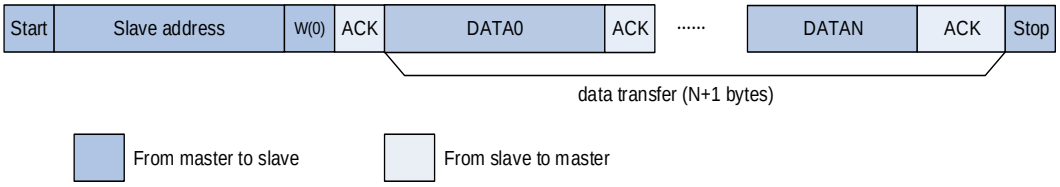
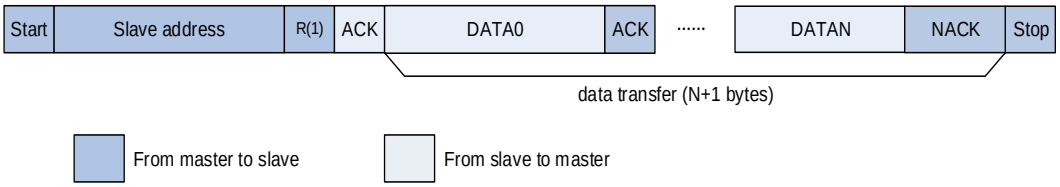


Figure 21-19. I2C communication flow with 7-bit address (Master Receive)



In 10-bit addressing mode, the HEAD10R bit can configured to decide whether the complete address sequence must be executed, or only the header to be sent. When HEAD10R=0, the complete 10 bit address read sequence must be excuted with START + header of 10-bit address in write direction + slave address byte 2 + RESTART + header of 10-bit address in read direction, as is shown in ***Figure 21-20. I2C communication flow with 10-bit address (Master Receive when HEAD10R=0)***.

In 10-bit addressing mode, if the master reception follows a master transmission between the same master and slave, the address read sequence can be RESTART + header of 10-bit address in read direction, as is shown in ***Figure 21-21. I2C communication flow with 10-bit address (Master Receive when HEAD10R=1)***.

Figure 21-20. I2C communication flow with 10-bit address (Master Receive when HEAD10R=0)

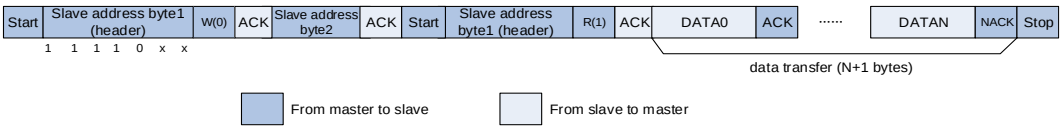
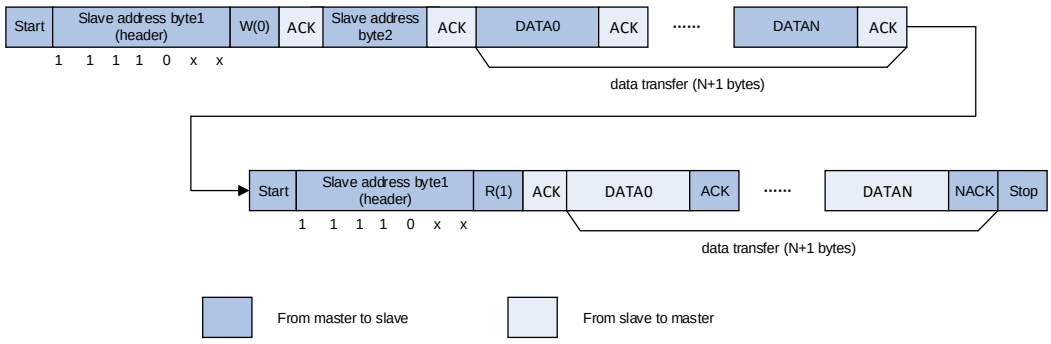


Figure 21-21. I2C communication flow with 10-bit address (Master Receive when HEAD10R=1)



Noise filter

Analog noise filter and digital noise filter are integrated in I2C peripherals, the noise filters can

be configured before the I2C peripheral is enabled according to the actual requirements.

The analog noise filter is disabled by setting the ANOFF bit in I2C_CTL0 register and enabled when ANOFF is 0. It can suppress spikes with a pulse width up to 50ns in fast mode and fast mode plus.

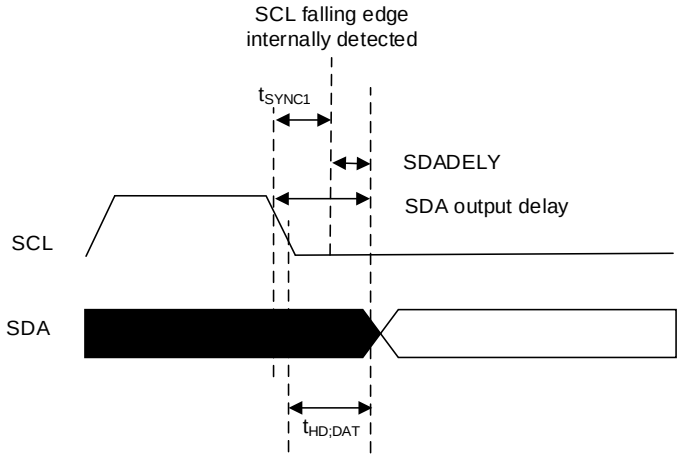
The digital noise filter can be used by configuring the DNF[3:0] bit in I2C_CTL0 register. The level of the SCL or the SDA will be changed if the level is stable for more than DNF[3:0]×t_{I2CCLK}. The length of spikes to be suppressed is configured by DNF[3:0].

I2C timings configuration

The PSC[3:0], SCLDELY[3:0] and SDADELY[3:0] bits in the I2C_TIMING register must be configured in order to guarantee a correct data hold and setup time used in I2C communication.

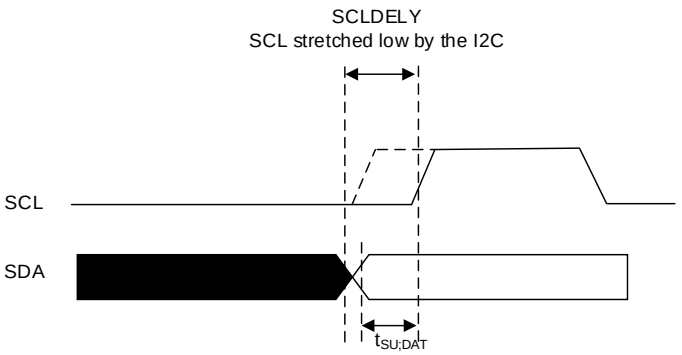
If the data is already available in I2C_TDATA register, the data will be sent on SDA after the SDADELY delay. As is shown in ***Figure 21-22. Data hold time***.

Figure 21-22. Data hold time



The SCLDELY counter starts when the data is sent on SDA output. As is shown in ***Figure 21-23. Data setup time***.

Figure 21-23. Data setup time



When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is $t_{SDADELY}$=SDADELY*t_{PSC}+t_{I2CCLK} where t_{PSC}=(PSC+1)*t_{I2CCLK}. $t_{SDADELY}$ effects $t_{HD;DAT}$. The total delay of SDA output is t_{SYNC1}+{[SDADELY*(PSC+1)+1]*t_{I2CCLK}}. t_{SYNC1} depends on SCL falling slope, the delay of analog filter, the delay of digital filter and delay of SCL synchronization to I2CCLK clock. The delay of SCL synchronization to I2CCLK clock is 2 to 3 t_{I2CCLK}.

SDADELY must match condition as follows:

- SDADELY≥{t_f(max)+$t_{HD;DAT}$(min)-t_{AF}(min)-[(DNF+3)*t_{I2CCLK}]}/[(PSC+1)*t_{I2CCLK}]
- SDADELY≤{$t_{HD;DAT}$(max)-t_{AF}(max)-[(DNF+4)*t_{I2CCLK}]}/[(PSC+1)*t_{I2CCLK}]

Note: t_{AF} is the delay of analog filter. The $t_{HD;DAT}$ should be less than the maximum of $t_{VD;DAT}$.

When SS = 0, after $t_{SDADELY}$ delay, the slave had to stretch the clock before the data writing to I2C_TDATA register, SCL is low during the data setup time. The setup time is $t_{SCLDELY}$=(SCLDELY+1)*t_{PSC}. $t_{SCLDELY}$ effects $t_{SU;DAT}$.

SCLDELY must match condition as follows:

- SCLDELY≥{[t_r(max)+$t_{SU;DAT}$(min)]/[(PSC+1)*t_{I2CCLK}]}-1

In master mode, the SCL clock high and low levels must be configured by programming the PSC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMING register.

When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is t_{SCLL}=(SCLL+1)*t_{PSC} where t_{PSC}=(PSC+1)*t_{I2CCLK}. t_{SCLL} impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is t_{SCLH}=(SCLH+1)*t_{PSC} where t_{PSC}=(PSC+1)*t_{I2CCLK}. t_{SCLH} impacts the SCL high time t_{HIGH}.

Note: When the I2C is enabled, the timing configuration and SS mode must not be changed.

Table 21-5. Data setup time and data hold time

Symbol	Parameter	Standard mode		Fast mode		Fast mode plus		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	us
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rising time of SCL and SDA	-	1000	-	300	-	120	-	1000	
t_f	falling time of SCL and SDA	-	300	-	300	-	120	-	300	

I2C reset

A software reset can be performed by clearing the I2CEN bit in the I2C_CTL0 register. When a software reset is generated, the SCL and SDA are released. The communication control bits and status bits come back to the reset value. Software reset have no effect on configuration registers. The impacted register bits are START, STOP, NACKEN in I2C_CTL1 register, I2CBSY, TBE, TI, RBNE, ADDSEND, NACK, TCR, TC, STPDET, BERR, LOSTARB and OUERR in I2C_STAT register. Additionally, when the SMBus is supported, PECTRANS in I2C_CTL1 register, PECERR, TIMEOUT and SMBALT in I2C_STAT are also impacted.

In order to perform the software reset, I2CEN must be kept low during at least 3 APB clock cycles. This is ensured by writing software sequence as follows:

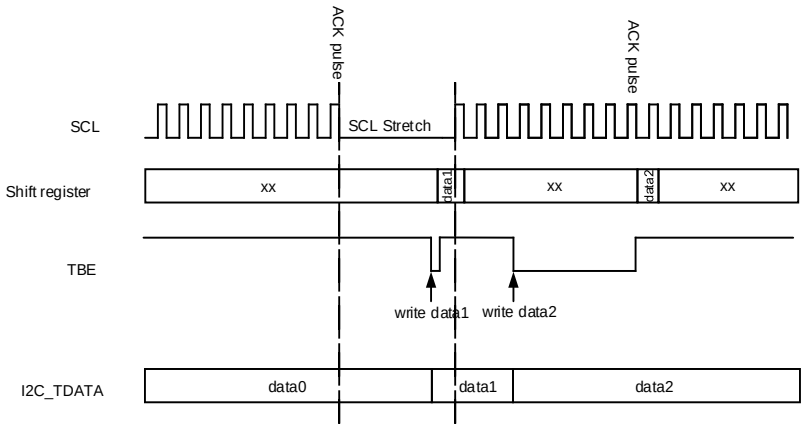
- Write I2CEN = 0
- Check I2CEN = 0
- Write I2CEN = 1

Data transfer

- **Data Transmission**

When transmitting data, if TBE is 0, it indicates that the I2C_TDATA register is not empty, the data in I2C_TDATA register is moved to the shift register after the 9th SCL pulse. Then the data will be transmitted through the SDA line from the shift register. If TBE is 1, it indicates that the I2C_TDATA register is empty, the SCL line is stretched low until I2C_TDATA is not empty. The stretch begins after the 9th SCL pulse.

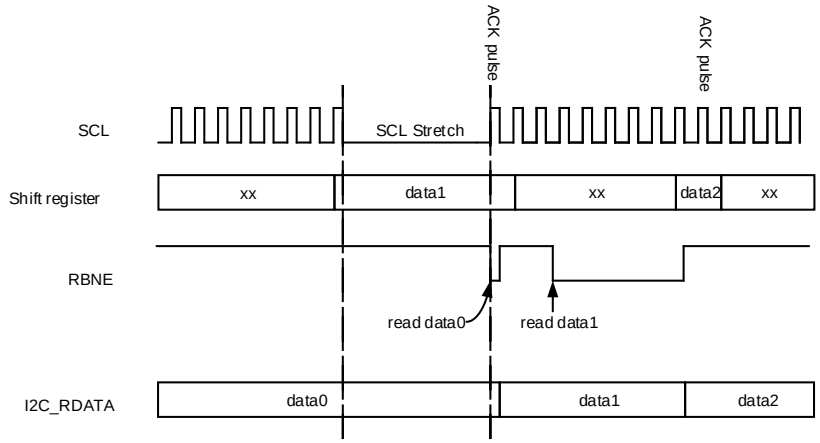
Figure 21-24. Data transmission



- **Data Reception**

When receiving data, the data will be received in the shift register first. If RBNE is 0, the data in the shift register will move into I2C_RDATA register. If RBNE is 1, the SCL line wii be stretched until the previous received data in I2C_RDATA is read. The stretch is inserted before the acknowledge pulse.

Figure 21-25. Data reception



- **Reload and automatic end mode**

In order to manage byte transfer and to shut down the communication in modes as is shown in ***Table 21-6. Communication modes to be shut down***, the I2C embedded a byte counter in the hardware.

Table 21-6. Communication modes to be shut down

Working mode	Action
Master mode	NACK, STOP and RESTART generation
Slave receiver mode	ACK control
SMBus mode	PEC generation/checking

The number of bytes to be transferred is configured by BYTENUM[7:0] in I2C_CTL1 register. If BYTENUM is greater than 255, or in slave byte control mode, the reload mode must be enabled by setting the RELOAD bit in I2C_CTL1 register. In reload mode, when BYTENUM counts to 0, the TCR bit will be set, and an interrupt will be generated if TCIE is set. Once the TCR flag is set, SCL is stretched. The TCR bit is cleared by writing a non-zero number in BYTENUM.

Note: The reload mode must be disabled after the last reloading of BYTENUM[7:0].

The reload mode must be disabled when the automatic end mode is enabled. In automatic end mode, the master will send a STOP signal automatically when the BYTENUM[7:0] counts to 0.

When reload mode and automatic end mode are disabled, the I2C communication process needs to be terminated by software. If the number of bytes in BYTENUM[7:0] has been transferred, the STOP bit should be set by software to generate a STOP signal, and then TC flag must be cleared.

I2C slave mode

- **Initialization**

When works in slave mode, at least one slave address should be enabled. Slave address 1 can be programmed in I2C_SADDR0 register and slave address 2 can be programmed in I2C_SADDR1 register. ADDRESSEN in I2C_SADDR0 register and ADDRESS2EN in I2C_SADDR1 register should be set when the corresponding address is used. 7-bit address or 10-bit address can be programmed in ADDRESS[9:0] in I2C_SADDR0 register by configuring the ADDFORMAT bit in 7-bit address or 10-bit address.

The ADDM[6:0] in I2C_CTL2 register defines which bits of ADDRESS[7:1] are compared with an incoming address byte, and which bits are ignored.

The ADDMSK2[2:0] is used to mask ADDRESS2[7:1] in I2C_SADDR1 register. For details, refer to the description of ADDMSK2[2:0] in I2C_SADDR1 register.

When the I2C received address matches one of its enabled addresses, the ADDSEND will be set, and an interrupt is generated if the ADDMIE bit is set. The READDR[6:0] bits in I2C_STAT register will store the received address. And TR bit in I2C_STAT register updates after the ADDSEND is set. The bit will let the slave to know whether to act as a transmitter or receiver.

- **SCL line stretching**

The clock stretching is used in slave mode by default (SS=0), the SCL line can be stretched low if necessary. The SCL will be stretched in following cases.

- The SCL is stretched when the ADDSEND bit is set, and released when the ADDSEND bit is cleared.
- In slave transmitting mode, after the ADDSEND bit is cleared, the SCL will be stretched before the first data byte writing to the I2C_TDATA register. Or the SCL will be stretched before the new data is written to the I2C_TDATA register after the previous data transmission is completed.
- In slave receiving mode, a new reception is completed but the data in I2C_RDATA register has not been read.
- When SBCTL=1 and RELOAD=1, after the transfer of the last byte, TCR is set. Before the TCR is cleared, the SCL will be stretched.
- The I2C stretches SCL low during $[(SDADELY+SCLDELY+1)*(PSC+1)+1]*t_{I2CCLK}$ after detecting the SCL falling edge.

The clock stretching can be disabled by setting the SS bit in I2C_CTL0 register (SS=1). The SCL will not be stretched in following cases.

- When the ADDSEND is set, the SCL will be not stretched.
- In slave transmitting mode, before the first SCL pulse, the data must be written in the I2C_TDATA register . Or else the OUERR bit in the I2C_STAT register will be set, if the ERRIE bit is set, an interrupt will be generated. When the STPDET bit is set and the first data transmission starts, OUERR bit in the I2C_STAT register will also be set.
- In slave receiving mode, before the 9th SCL pulse (ACK pulse) occurred by the next data byte, the data must be read out from the I2C_RDATA register. Or else the OUERR bit in the I2C_STAT register will be set, if the ERRIE bit is set, and an interrupt will be generated.

- **Slave byte control mode**

In slave receiving mode, the slave byte control mode can be enabled by setting the SBCTL bit in the I2C_CTL0 register to allow byte ACK control. When SS=1, the slave byte control mode is not allowed.

When using slave byte control mode, the reload mode must be enabled by setting the RELOAD bit in I2C_CTL1 register. In slave byte control mode, BYTENUM[7:0] in I2C_CTL1 register must be configured as 1 in the ADDSEND interrupt service routine and reloaded to 1 after each byte received. The TCR bit in I2C_STAT register will be set when a byte is received, the SCL will be stretched low by slave between the 8th and 9th clock pulses. Then the data can be read from the I2C_RDATA register, and the slave determines to send an ACK or a NACK by configuring the NACKEN bit in the I2C_CTL1 register. When the BYTENUM[7:0] is written a non-zero value, the slave will release the stretch.

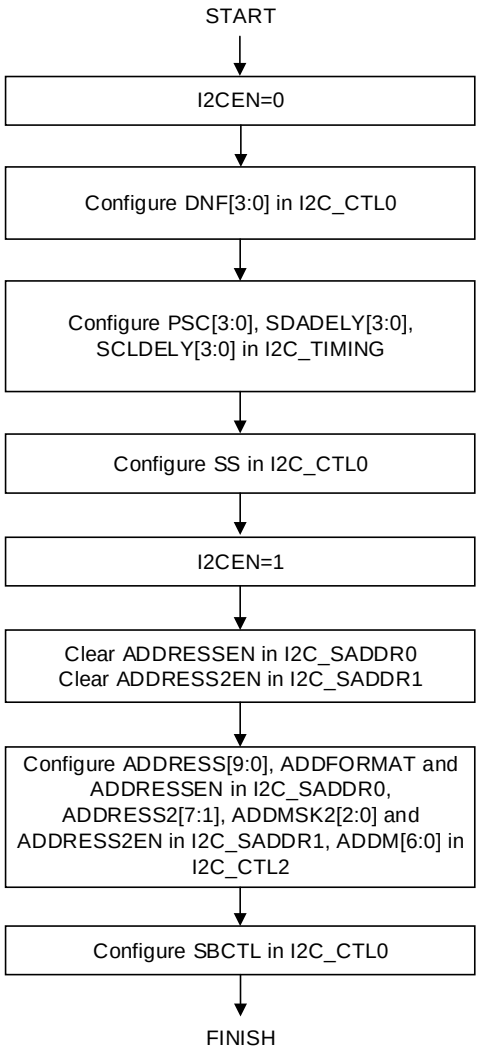
When the BYTENUM[7:0] is greater than 0x1, there is no stretch between the reception of two data bytes.

Note: The SBCTL bit can be configured in following cases:

1. I2CEN=0.
2. The slave has not been addressed.
3. ADDSEND=1.

Only when the ADDSEND=1, or TCR=1, the RELOAD bit can be modified.

Figure 21-26. I2C initialization in slave mode



- **Programming model in slave transmitting mode**

When the I2C_TDATA register is empty, the TI bit in I2C_STAT register will be set. If the TIE bit in I2C_CTL0 register is set, an interrupt will be generated. The NACK bit in I2C_STAT register will be set when a NACK is received. And an interrupt is generated if the NACKIE bit is set in the I2C_CTL0 register. The TI bit in I2C_STAT register will not be set when a NACK is received.

The STPDET bit in I2C_STAT register will be set when a STOP is received. If the STPDETIE in I2C_CTL0 register is set, an interrupt will be generated.

When SBCTL is 0, if ADDSEND=1, and the TBE bit in I2C_STAT register is 0, the data in I2C_TDATA register can be chosed to be transmitted or flushed. The data is flushed by setting the TBE bit.

When SBCTL=1, the slave works in slave byte control mode, the BYTENUM[7:0] must be configured in the ADDSEND interrupt service routine. And the number of TI events is equal to the value of BYTENUM[7:0].

When SS=1, the SCL will not be stretched when ADDSEND bit in I2C_STAT register is set. In this case, the data in I2C_TDATA register can not be flushed in ADDSEND interrupt service routine. So the first byte to be sent must be programmed in the I2C_TDATA register previously.

- This data can be the one which is written in the last TI event of the last transfer.
- Setting the TBE bit can flush the data if it is not the one to be sent, then a new byte can be written in I2C_TDATA register. The STPDET must be 0 when the data transmission begins. Or else the OUERR bit in I2C_STAT register will be set and an underrun error occurs.
- When interrupt or DMA is used in slave transmitter, if a TI event is needed, in order to generate a TI event both the TI bit and the TBE bit must be set.

Figure 21-27. Programming model for slave transmitting when SS=0

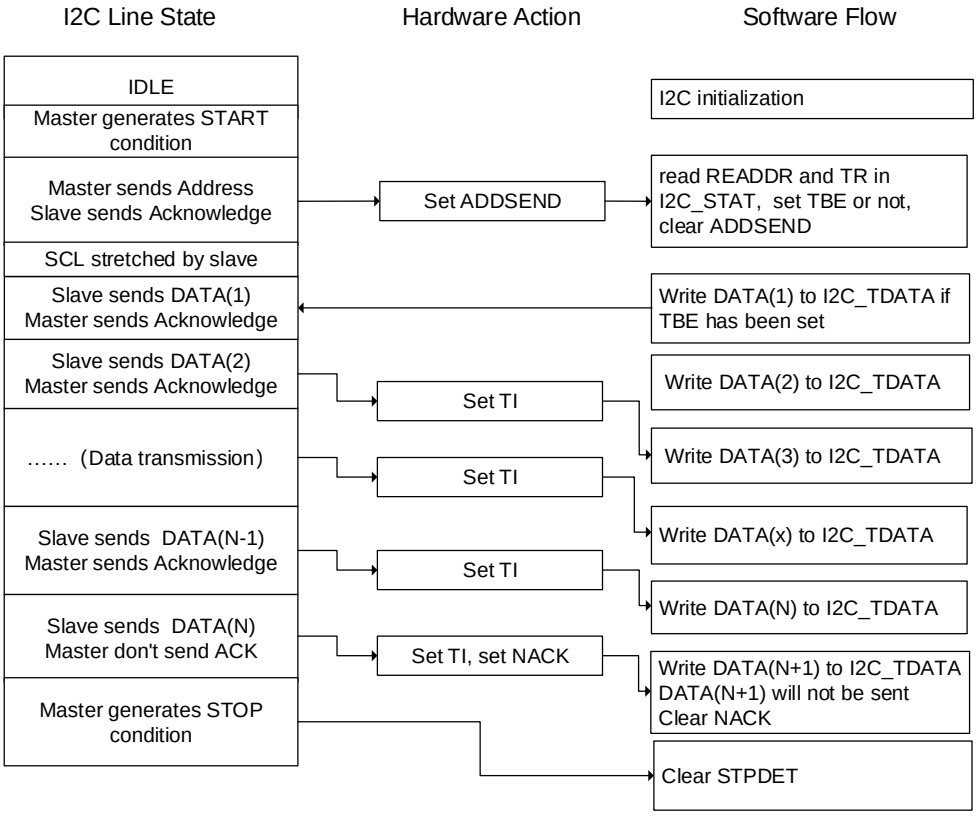
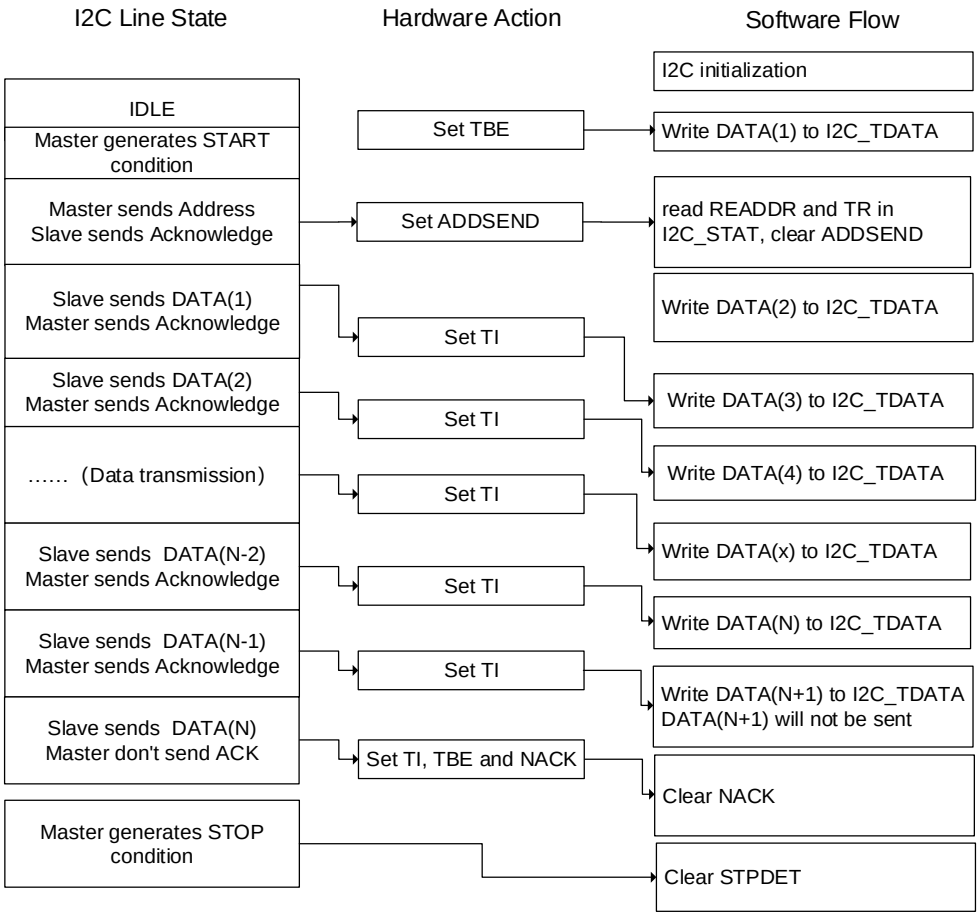


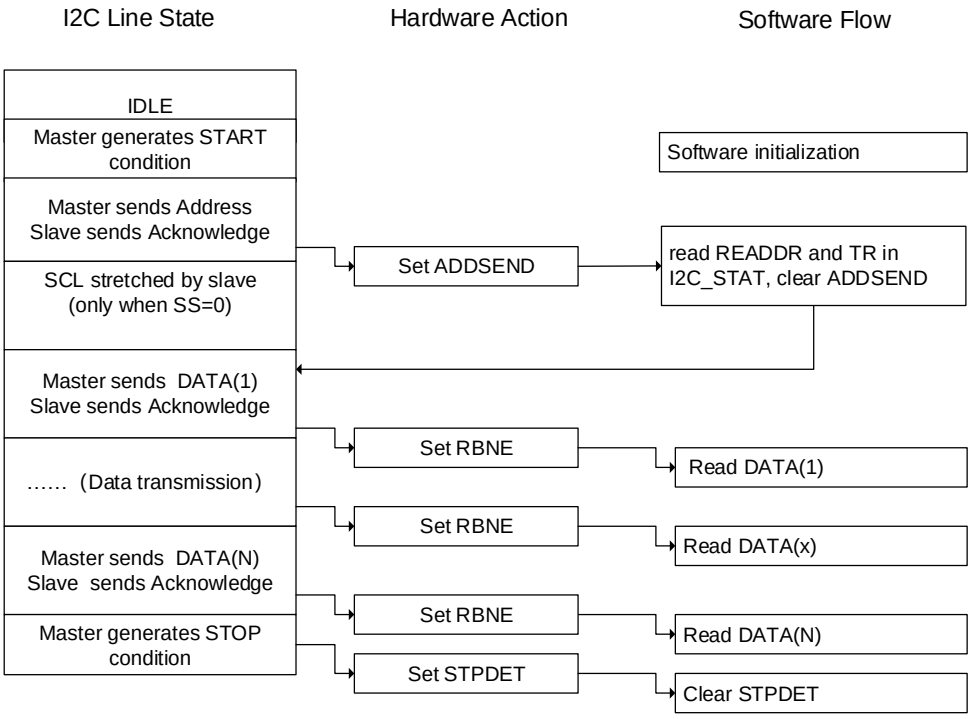
Figure 21-28. Programming model for slave transmitting when SS=1



- **Programming model in slave receiving mode**

When the I2C_RDATA is not empty, the RBNE bit in I2C_STAT register is set, and if the RBNEIE bit in I2C_CTL0 register is set, an interrupt will be generated. When a STOP is received, STPDET will be set in I2C_STAT register. If the STPDETIE bit in I2C_CTL0 register is set, and an interrupt will be generated.

Figure 21-29. Programming model for slave receiving



I2C master mode

Initialization

The SCLH[7:0] and SCLL[7:0] in I2C_TIMING register should be configured when I2CEN is 0. In order to support multi-master communication and slave clock stretching, a clock synchronization mechanism is implemented.

The SCLL[7:0] and SCLH[7:0] are used for the low level counting and high level couting respectively. After a t_{SYNC1} delay, when the SCL low level is detected, the SCLL[7:0] starts counting, if the SCLL[7:0] in I2C_TIMING register is reached by SCLL[7:0] counter, the I2C will release the SCL clock. After a t_{SYNC2} delay, when the SCL high level is detected, the SCLH[7:0] starts counting, if the SCLH[7:0] in I2C_TIMING register is reached by SCLH[7:0] counter, the I2C will stretch the SCL clock.

So the master clock period is:

$t_{SCL}=t_{SYNC1}+t_{SYNC2}+\{[(SCLH[7:0]+1)+(SLLL[7:0]+1)]*(PSC+1)*t_{I2CCLK}\}$.

The t_{SYNC1} depends on the SCL falling slope, delay by input analog and digital noise filter and SCL synchronization with I2CCLK clock, which generally 2 to 3 I2CCLK periods. The t_{SYNC2} depends on the SCL rising slope, delay by input analog and digital noise filter and SCL synchronization with I2CCLK clock, which generally 2 to 3 I2CCLK periods. The delay by digital noise filter is $DNF[3:0]*t_{I2CCLK}$.

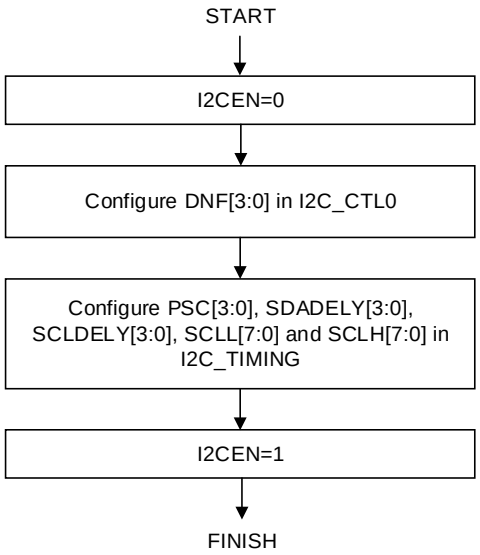
When works in master mode, the ADD10EN bit, SADDRESS[9:0] bits, TRDIR bit should be configured in I2C_CTL1 register. When the addressing mode is 10-bit in master receiving

mode, the HEAD10R bit must be configured to decide whether the complete address sequence must be executed, or only the header to be sent. The number of bytes to be transferred should be configured in BYTENUM[7:0] in I2C_CTL1 register. If the number of bytes to be transferred is equal to or greater than 255, BYTENUM[7:0] should be configured as 0xFF. Then the master sends the START signal. All the bits above should be configured before the START is set. The slave address will be sent after the START signal when the I2CBSY bit I2C_STAT register is detected as 0. When the arbitration is lost, the master changes to slave mode and the START bit will be cleared by hardware. When the slave address has been sent, the START bit will be cleared by hardware.

In 10-bit addressing mode, if the master receives a NACK after the transmission of 10-bit header, the master will resend it until ACK is received. The ADDSENDC bit must be set to stop sending the slave address.

If the START bit is set, meanwhile the ADDSEND is set by addressing as a slave, the master changes to slave mode. The ADDSENDC bit must be set to clear the START bit.

Figure 21-30. I2C initialization in master mode



- **Programming model in master transmitting mode**

In master transmitting mode, the TI bit is set after the ACK is received of each byte transmission. If the TIE bit in I2C_CTL0 register is set, an interrupt will be generated. The bytes to be transferred is programmed in BYTENUM[7:0] in I2C_CTL0 register. If the bytes to be transferred is greater than 255, RELOAD bit in I2C_CTL0 register must be set to enable the reload mode. In reload mode, when data of BYTENUM[7:0] bytes have been transferred, the TCR bit in I2C_STAT register will be set and the SCL stretches unitil BYTENUM[7:0] is modified with a non-zero value.

When a NACK is received, the TI bit will not set.

- If data of BYTENUM[7:0] bytes have been transferred and RELOAD=0, the AUTOEND

bit in I2C_CTL1 can be set to generate a STOP signal automatically. When AUTOEND is 0, the TC bit in I2C_STAT register will be set and the SCL is stretched. In this case, the master can generate a STOP signal by setting the STOP bit in the I2C_CTL1 register. Or generate a RESTART signal to start a new transfer. The TC bit is cleared when the START / STOP bit is set.

- If a NACK is received, a STOP signal is automatically generated, the NACK is set in I2C_STAT register, if the NACKIE bit is set, an interrupt will be generated.

Note: When the RELOAD bit is 1, the AUTOEND has no effect.

Figure 21-31. Programming model for master transmitting (N<=255)

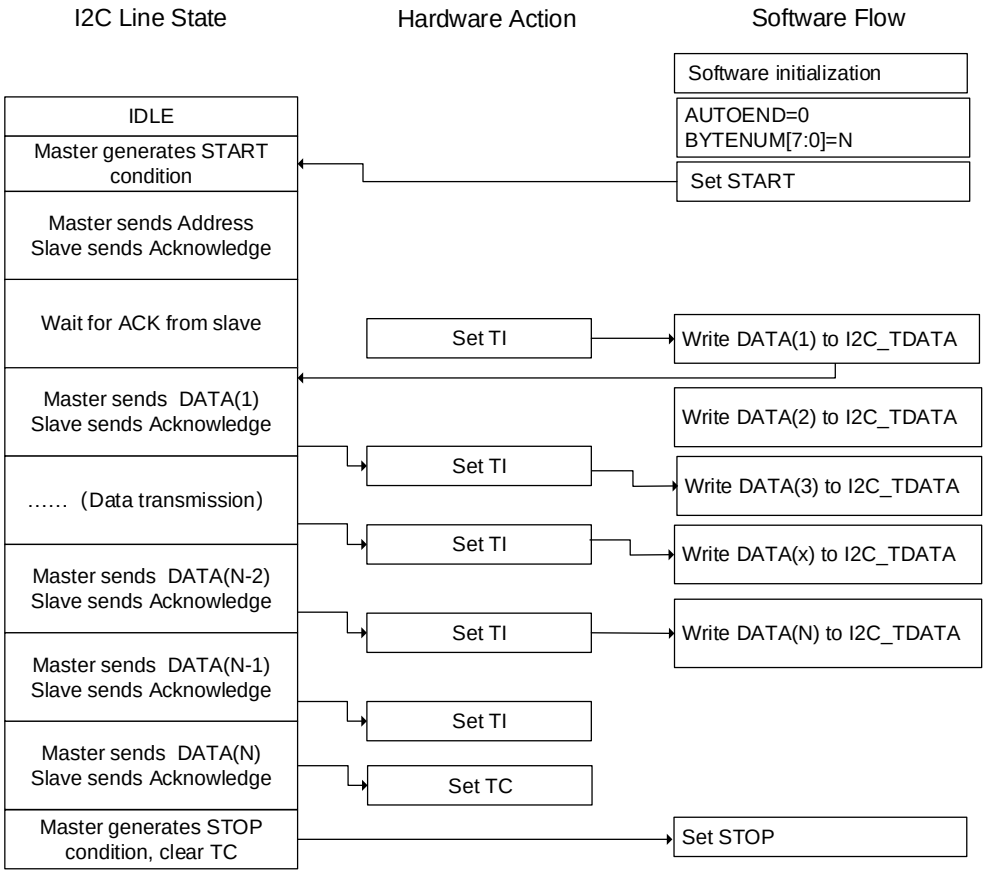
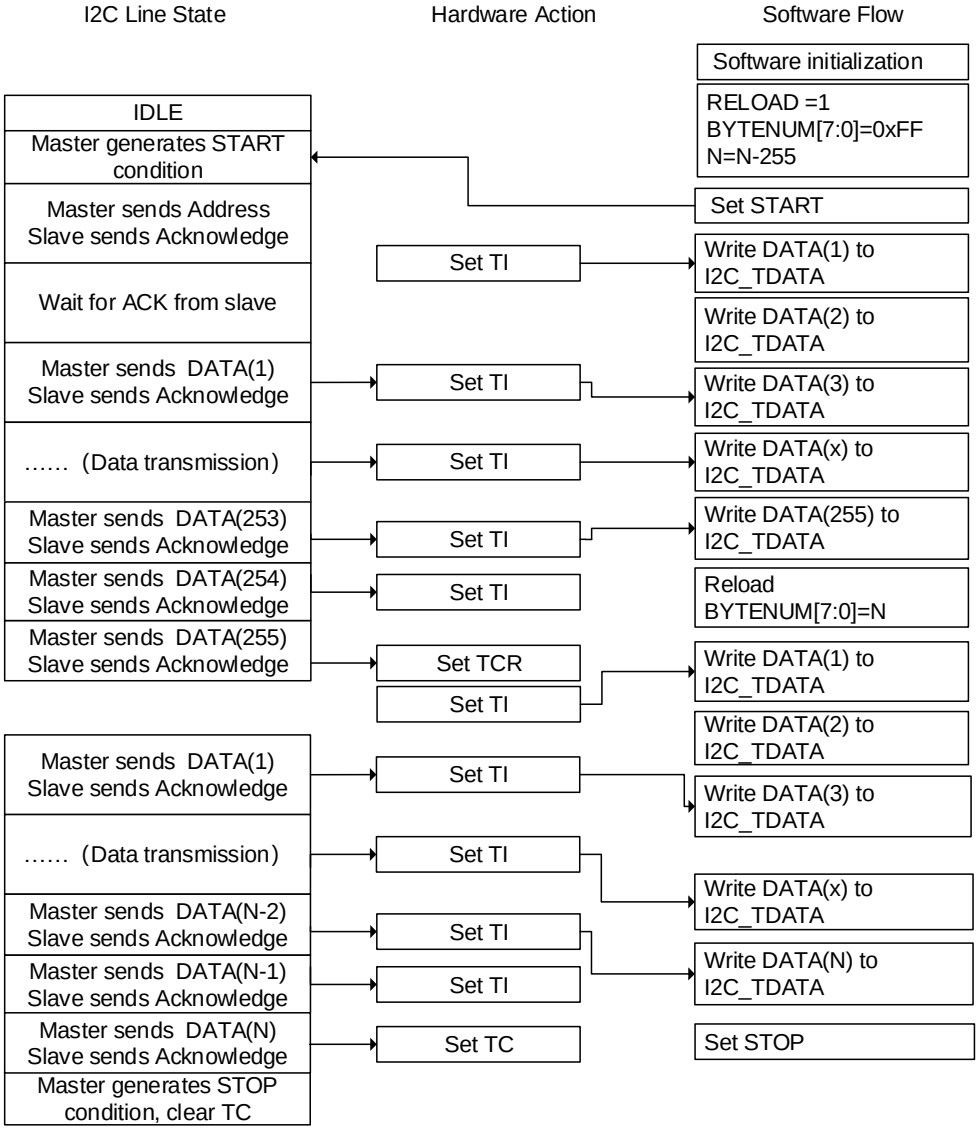


Figure 21-32. Programming model for master transmitting (N>255)



- **Programming model in master receiving mode**

In master receiving mode, the RBNE bit in I2C_STAT register will be set when a byte is received. If the RBNEIE bit is set in I2C_CTL0 register, an interrupt will be generated. If the number of bytes to be received is greater than 255, RELOAD bit in I2C_CTL0 register must be set to enable the reload mode. In reload mode, when data of BYTENUM[7:0] bytes have been transferred, the TCR bit in I2C_STAT register will be set and the SCL stretches unitil BYTENUM[7:0] is modified with a non-zero value.

If data of BYTENUM[7:0] bytes have been transferred and RELOAD=0, the AUTOEND bit in I2C_CTL1 can be set to generate a STOP signal automatically. When AUTOEND is 0, the TC bit in I2C_STAT register will be set and the SCL is stretched. In this case, the master can generate a STOP signal by setting the STOP bit in the I2C_CTL1 register. Or generate a RESTART signal to start a new transfer. The TC bit is cleared when the START / STOP bit is set.

Figure 21-33. Programming model for master receiving (N<=255)

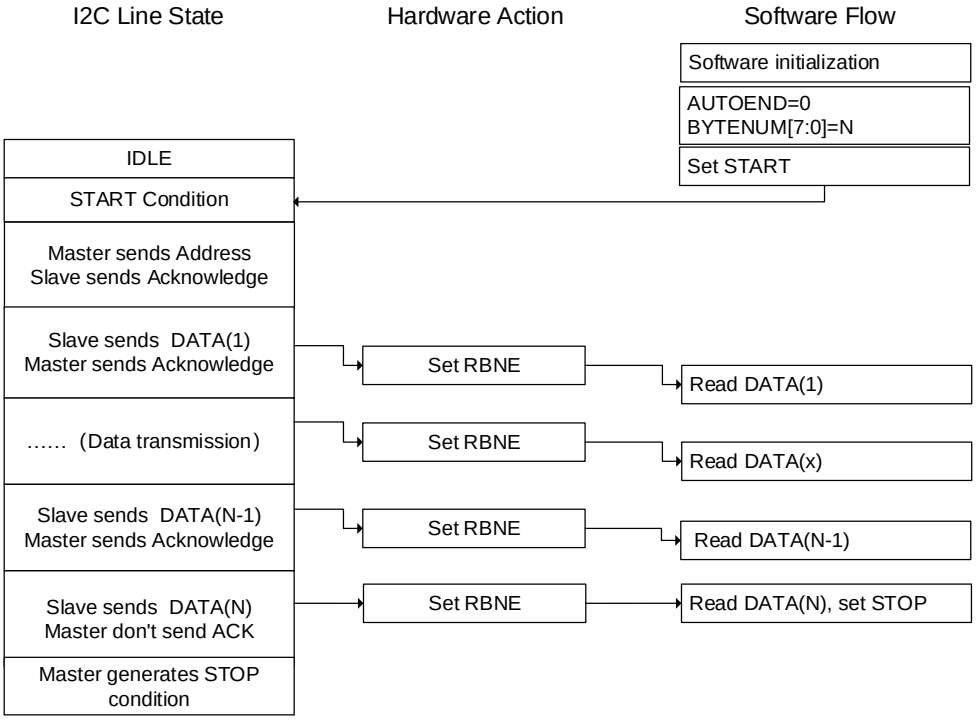
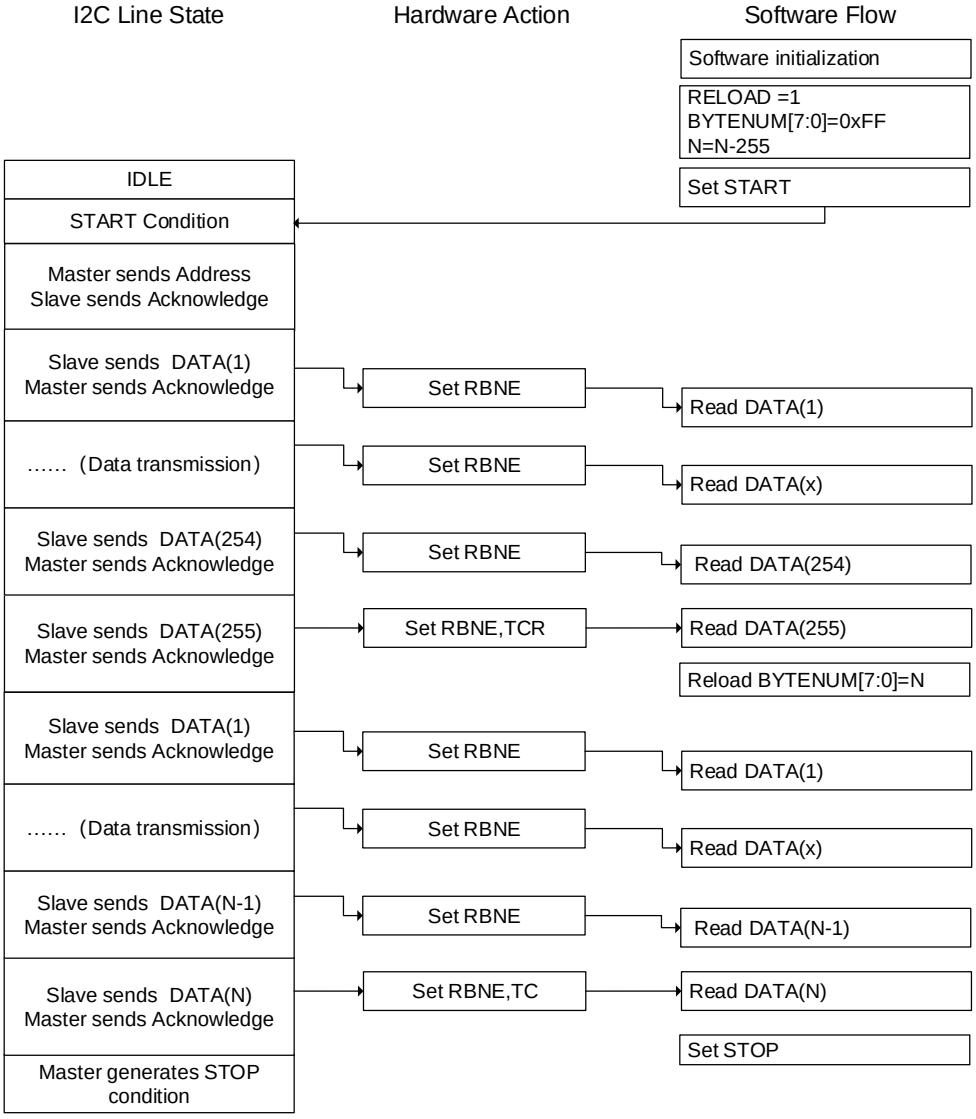


Figure 21-34. Programming model for master receiving (N>255)



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 this protocol there is a very useful distinction between a system host and all the other devices in the system, that is the host provides address assignment function.

- **SMBus slave byte control**

The slave byte control of SMBus receiver is the same as I2C. It allows the ACK control of each byte.The Slave Byte Control mode must be enabled by setting SBCTL bit in I2C_CTL0 register.

- **Host notify protocol**

When the SMBHAEN bit in the I2C_CTL0 register is set, the SMBus supports the host notify protocol. In this protocol, the device acts as a master and the host as a slave, and the host will acknowledge the SMBus host address.

- **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 25~35ms. 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.

The timeout detection can be enabled by setting TOEN and EXTOEN bits in the I2C_TIMEOUT register. The timer must be configured to guarantee that the timeout detected before the maximum time given in the SMBus specification.

The value programmed in BUSTOA[11:0] is used to check the $t_{TIMEOUT}$ parameter. To detect the SCL low level timeout, the TOIDLE bit must be 0. And the timer can be enabled by setting the TOEN bit in the I2C_TIMEOUT register, after the TOEN bit is set, the BUSTOA[11:0] and the TOIDLE bit cannot be changed. If the low level time of SCL is greater than (BUSTOA+1)*2048*t_{I2CCLK}, the TIMEOUT flag will be set in I2C_STAT register.

The BUSTOB[11:0] is used to check the $t_{LOW:SEXT}$ of the slave and the $t_{LOW:MEXT}$ of the master. The timer can be enabled by setting the EXTOEN bit in the I2C_TIMEOUT register, after the EXTOEN bit is set, the BUSTOB[11:0] cannot be changed. If the SCL stretching time

of the SMBus peripheral is greater than (BUSTOB+1)*2048*t_{I2CCLK} and within the timeout interval described in the bus idle detection section, the TIMEOUT bit in the I2C_STAT register will be set.

- **Packet error checking**

There is a CRC-8 calculator in I2C block to perform Packet Error Checking for I2C data. A PEC (packet error code) byte is appended at the end of each transfer. The byte is calculated as CRC-8 checksum, calculated over the entire message including the address and read/write bit. The polynomial used is x^8+x^2+x+1 (the CRC-8-ATM HEC algorithm, initialized to zero).

When I2C is disabled, the PEC can be enabled by setting the PECEN bit in I2C_CTL0 register.Since the PEC transmission is managed by BYTENUM[7:0] in I2C_CTL1 register, SBCTL bit must be set when act as a slave. When PECTRANS is set and the RELOAD bit is cleared, PEC is transmitted after the BYTENUM[7:0]-1 data byte. The PECTRANS has no effect if RELOAD is set.

- **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. The host processes the interrupt and accesses all SMBALERT# devices through the Alert Response Address at the same time. If the SMBALERT# is pulled low by the devices, the devices will acknowledge the Alert Response Address. When SMBHAEN is 0, it is configured as a slave device, the SMBA pin will be pulled low by setting the SMBALTEN bit in the I2C_CTL0 register. Meanwhile the Alert Response Address is enabled. When SMBHAEN is 1, it is configured as a host, and the SMBALTEN is 1, as soon as a falling edge is detected on the SMBA pin, the SMBALT flag will be set in the I2C_STAT register. If the ERRIE bit is set in the I2C_CTL0 register, an interrupt will be generated. When SMBALTEN is 0, the level of ALERT line is considered high even if the SMBA pin is low. The SMBA pin can be used as a standard GPIO if SMBALTEN is 0.

- **Bus idle detection**

If the master detects that the high level duration of the clock and data signals is greater than $t_{HIGH,MAX}$, the bus can be considered idle.

This timing parameter includes the case of a master that has been dynamically added to the bus and may not have detected a state transition on a SMBCLK or SMBDAT lines. In this case, in order to ensure that there is no ongoing transmission, the master must wait long enough.

The BUSTOA[11:0] bits must be programmed with the timer reload value to enable the t_{IDLE} check in order to obtain the t_{IDLE} parameter. To detect SCL and SDA high level timeouts, the TOIDLE bit must be set. Then setting the TOEN bit in the I2C_TIMEOUT register to enable the timer, after the TOEN bit is set, the BUSTOA[11:0] bit and the TOIDLE bit cannot be changed. If the high level time of both SCL and SDA is greater than (BUSTOA+1)*4*t_{I2CCLK}, the TIMEOUT flag will be set in the I2C_STAT register.

- **SMBus slave mode**

The SMBus receiver must be able to NACK each command or data it receives. For ACK control in slave mode, slave byte control mode can be enabled by setting SBCTL bit in I2C_CTL0 register.

SMBus-specific addresses should be enabled when needed. The SMBus Device Default address (0b1100 001) is enabled by setting the SMBDAEN bit in the I2C_CTL0 register. The SMBus Host address (0b0001 000) is enabled by setting the SMBHAEN bit in the I2C_CTL0 register. The Alert Response Address (0b0001 100) is enabled by setting the SMBALTEN bit in the I2C_CTL0 register.

SMBus mode

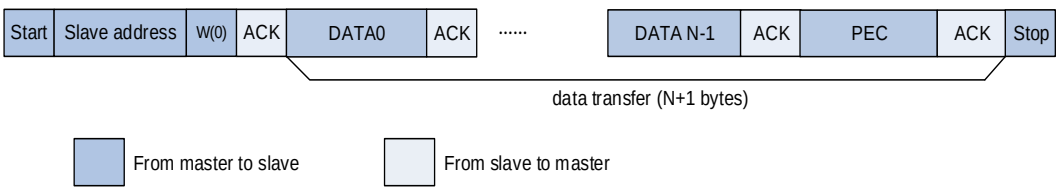
- **SMBus master transmitter and slave receiver**

The PEC in SMBus master mode can be transmitted by setting the PECTRANS bit before setting the START bit, and the number of bytes in the BYTENUM[7:0] field must be configured. In this case, the total number of transmissions when TI interrupt occur is BYTENUM-1. So if BYTENUM=0x1 and PECTRANS bit is set, the data in I2C_PEC register will be transmitted automatically. If AUTOEND is 1 the SMBus master will send the STOP signal after the PEC byte automatically. If the AUTOEND is 0, the SMBus master can send a RESTART signal after the PEC. The PEC byte in I2C_PEC register will be sent after BYTENUM-1 bytes, and the TC flag will be set after PEC is sent, then the SCL line is stretched low. The RESTART must be set in the TC interrupt routine.

When used as slave receiver, in order to allow PEC checking at the end of the number of bytes transmitted, SBCTL must be set. To configure ack control for each byte, the RELOAD must be set. In order to check the PEC byte, it is necessary to clear the RELOAD bit and set PECTRANS bit. After receiving BYTENUM-1 data, the next received byte will be compared with the data in the I2C_PEC register. If the PEC values does not match, the NACK is automatically generated. If the PEC values matches, the ACK is automatically generated, regardless of the NACKEN bit value. When PEC byte is received, it is also copied into the I2C_RDATA register, and RBNE flag will be set. If the ERRIE bit in I2C_CTL0 register is 1, when PEC value does not match, the PECERR flag will be set and the interrupt will be generated. If ACK control is not required, then PECTRANS can be set to 1 and BYTENUM can be programmed according to the number of bytes to be received.

Note: After the RELOAD bit is set, the PECTRANS cannot be changed.

Figure 21-35. SMBus master transmitter and slave receiver communication flow

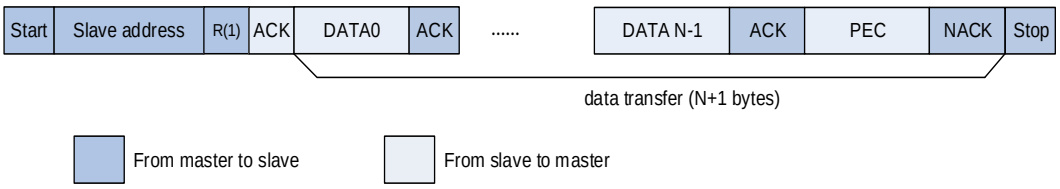


- **SMBus master receiver and slave transmitter**

- If the SMBus master is required to receive PEC at the end of bytes transfer, automatic end mode can be enabled. Before sending a START signal on the bus, PECTRANS bit must be set and slave addresses must be programmed. After receiving BYTENUM-1 data, the next received byte will be compared with the data in the I2C_PEC register automatically. A NACK is respond to the PEC byte before STOP signal.
- If the SMBus master receiver is required to generate a RESTART signal after receiving PEC byte, automatic end mode must be disabled. Before sending a START signal to the bus, PECTRANS bit must be set and slave addresses must be programmed. After receiving BYTENUM-1 data, the next received byte will be compared with the data in the I2C_PEC register automatically. The TC flag will be set after PEC is sent, then the SCL line is stretched low. The RESTART can be set in the TC interrupt routine.
- When used as slave transmitter, in order to allow PEC transmission at the end of BYTENUM[7:0] bytes, SBCTL must be set. If PECTRANS bit is set, the number of bytes in BYTENUM[7:0] contains PEC byte. In this case, if the number of bytes requested by the master is greater than BYTENUM-1, the total number of TI interrupts will be BYTENUM-1, and the data of the I2C_PEC register will be transmitted automatically.

Note: After the RELOAD bit is set, the PECTRANS cannot be changed.

Figure 21-36. SMBus master receiver and slave transmitter communication flow



Wakeup from power saving modes

When the address of I2C matches correctly, it can wake up MCU from Deep-sleep mode, Deep-sleep 1 mode and Deep-sleep 2 mode. In order to wake up from these power saving modes, WUEN bit must be set in the I2C_CTL0 register and the IRC8M must be selected as the clock source for I2CCLK. During Deep-sleep mode, Deep-sleep 1 mode and Deep-sleep 2 mode, the IRC8M is switched off. The I2C interface switches the IRC8M on, and stretches SCL low until IRC8M is woken up when a START is detected. Then the IRC8M is used as the clock of I2C to receive the address. When address matching is detected, I2C stretches SCL during MCU wake-up. The SCL is released until the software clears the ADDSEND flag and the transmission proceeds normally. If the detected address does not match, IRC8M will be closed again and the MCU will not be wake up.

Only an address match interrupt (ADDMIE=1) can wakeup the MCU. If the clock source of I2C is the system clock, or WUEN = 0, IRC8M will not switched on after receiving start signal. When wakeup from power saving mode is enabled, the digital filter must be disabled and the SS bit in I2C_CTL0 must be cleared. Before entering power saving mode, the I2C peripheral must be disabled (I2CEN=0) if wakeup from power saving mode is disabled (WUEN =0).

Use DMA for data transfer

As is shown in I2C slave mode and I2C master mode, each time TI 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 TI and RBNE flag: each time TI or RBNE is asserted, DMA controller does a read or write operation automatically.

The DMA transmission request is enabled by setting the DENT bit in I2C_CTL0 register. The DMA reception request is enabled by setting the DENR bit in I2C_CTL0 register. In master mode, the slave address, transmission direction, number of bytes and START bit are programmed by software. The DMA must be initialized before setting the START bit. The number of bytes to be transferred is configured in the BYTENUM[7:0] in I2C_CTL1 register. In slave mode, the DMA must be initialized before the address match event or in the ADDSEND interrupt routine, before clearing the ADDSEND flag.

I2C error and interrupts

The I2C error flags are listed in ***Table 21-7. I2C error flags***.

Table 21-7. I2C error flags

I2C Error Name	Description
BERR	Bus error
LOSTARB	Arbitration lost
OUERR	Overrun / Underrun flag
PECERR	CRC value doesn't match
TIMEOUT	Bus timeout in SMBus mode
SMBALT	SMBus Alert

The I2C interrupt events and flags are listed in ***Table 21-8. I2C interrupt events***.

Table 21-8. I2C interrupt events

Interrupt event	Event flag	Enable control bit
I2C_RDATA is not empty during receiving	RBNE	RBNEIE
Transmit interrupt	TI	TIE
STOP signal detected in slave mode	STPDET	STPDETIE
Transfer complete reload	TCR	TCIE
Transfer complete	TC	
Address match	ADDSEND	ADDMIE
Not acknowledge received	NACK	NACKIE
Bus error	BERR	ERRIE
Arbitration Lost	LOSTARB	
Overrun / Underrun error	OUERR	
PEC error	PECERR	
Timeout error	TIMEOUT	
SMBus Alert	SMBALT	

I2C debug mode

When the microcontroller enters the debug mode (Cortex®-M33 core halted), the SMBus timeout either continues to work normally or stops, depending on the I2Cx_HOLD configuration bits in the DBG module.

21.2.4. Register definition

I2C2 base address: 0x4000 C000

Control register 0 (I2C_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								PECEN	SMBALTEN	SMBDAEN	SMBHAEN	GCEN	WUEN	SS	SBCTL
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DENR	DENT	Reserved	ANOFF	DNF[3:0]				ERRIE	TCIE	STPDETIE	NACKIE	ADDMIE	RBNEIE	TIE	I2CEN
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	PECEN	PEC Calculation Switch 0: PEC Calculation off 1: PEC Calculation on
22	SMBALTEN	SMBus Alert enable 0: SMBA pin is not pulled down (device mode) or SMBus Alert pin SMBA is disabled (host mode) 1: SMBA pin is pulled down (device mode) or SMBus Alert pin SMBA is enabled (host mode)
21	SMBDAEN	SMBus device default address enable 0: Device default address is disabled, the default address 0b1100001x will be not acknowledged. 1: Device default address is enabled, the default address 0b1100001x will be acknowledged.
20	SMBHAEN	SMBus host address enable 0: Host address is disabled, address 0b0001000x will be not acknowledged. 1: Host address is enabled, address 0b0001000x will be acknowledged.
19	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
18	WUEN	Wakeup from power saving mode enable, including Deep-sleep mode, Deep-sleep

		1 mode and Deep-sleep 2 mode. 0: Wakeup from power saving mode disable. 1: Wakeup from power saving mode enable. **Note:** WUEN can be set only when DNF[3:0] = 0000
17	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 **Note:** When in master mode, this bit must be 0. This bit can be modified when I2CEN = 0.
16	SBCTL	Slave byte control This bit is used to enable hardware byte control in slave mode. 0: Slave byte control is disabled 1: Slave byte control is enabled
15	DENR	DMA enable for reception 0: DMA is disabled for reception 1: DMA is enabled for reception
14	DENT	DMA enable for transmission 0: DMA is disabled for transmission 1: DMA is enabled for transmission
13	Reserved	Must be kept at reset value.
12	ANOFF	Analog noise filter disable 0: Analog noise filter is enabled 1: Analog noise filter is disabled **Note:** This bit can only be programmed when the I2C is disabled (I2CEN = 0).
11:8	DNF[3:0]	Digital noise filter 0000: Digital filter is disabled 0001: Digital filter is enabled and filter spikes with a length of up to 1 t_{I2CCLK} ... 1111: Digital filter is enabled and filter spikes with a length of up to15 t_{I2CCLK} These bits can only be modified when the I2C is disabled (I2CEN = 0).
7	ERRIE	Error interrupt enable 0: Error interrupt disabled 1: Error interrupt enabled. When BERR, LOSTARB, OUERR, PECERR, TIMEOUT or SMBALT bit is set, an interrupt will be generated.
6	TCIE	Transfer complete interrupt enable 0: Transfer complete interrupt is disabled 1: Transfer complete interrupt is enabled
5	STPDETIE	Stop detection interrupt enable

		0: Stop detection (STPDET) interrupt is disabled 1: Stop detection (STPDET) interrupt is enabled
4	NACKIE	NACK received interrupt enable 0: NACK received interrupt is disabled 1: NACK received interrupt is enabled
3	ADDMIE	Address match interrupt enable in slave mode 0: Address matchinterrupt is disabled 1: Address matchnterrupt is enabled
2	RBNEIE	Receive interrupt enable 0: Receive (RBNE) interrupt is disabled 1: Receive (RBNE) interrupt is enabled
1	TIE	Transmit interrupt enable 0: Transmit (TI) interrupt is disabled 1: Transmit (TI) interrupt is enabled
0	I2CEN	I2C peripheral enable 0: I2C is disabled 1: I2C is enabled

Control register 1 (I2C_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					PECTRANS	AUTOEND	RELOAD	BYTENUM[7:0]							
					rw	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACKEN	STOP	START	HEAD10R	ADD10EN	TRDIR	SADDRESS[9:0]									
rw	rw	rw	rw	rw	rw	rw									

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26	PECTRANS	PEC Transfer Set by software. Cleared by hardware in the following cases: When PEC byte is transferred or ADDSEND bit is set or STOP signal is detected or I2CEN=0. 0: Don’t transfer PEC value

		1: Transfer PEC **Note:** This bit has no effect when RELOAD=1, or SBCTL=0 in slave mode.
25	AUTOEND	Automatic end mode in master mode 0: TC bit is set when the transfer of BYTENUM[7:0] bytes is completed. 1: a STOP signal is sent automatically when the transfer of BYTENUM[7:0] bytes is completed. **Note:** This bit works only when RELOAD=0. This bit is set and cleared by software.
24	RELOAD	Reload mode 0: After the data of BYTENUM[7:0] bytes transfer, the transfer is completed. 1: After data of BYTENUM[7:0] bytes transfer, the transfer is not completed and the new BYTENUM[7:0] will be reloaded. Every time when the BYTENUM[7:0] bytes have been transferred, the TCR bit in I2C_STAT register will be set. This bit is set and cleared by software.
23:16	BYTENUM[7:0]	Number of bytes to be transferred These bits are programmed with the number of bytes to be transferred. When SBCTL=0, these bits have no effect. **Note:** These bits should not be modified when the START bit is set.
15	NACKEN	Generate NACK in slave mode 0: an ACK is sent after receiving a new byte. 1: a NACK is sent after receiving a new byte. **Note:** The bit can be set by software, and cleared by hardware when the NACK is sent, or when a STOP signal is detected or ADDSEND is set, or when I2CEN=0. When PEC is enabled, whether to send an ACK or a NACK is not depend on the NACKEN bit. When SS=1, and the OUERR bit is set, the value of NACKEN is ignored and a NACK will be sent.
14	STOP	Generate a STOP signal on I2C bus This bit is set by software and cleared by hardware when I2CEN=0 or STOP signal is detected. 0: STOP will not be sent 1: STOP will be sent
13	START	Generate a START signal on I2C bus This bit is set by software and cleared by hardware after the address is sent. When the arbitration is lost, or a timeout error occurred, or I2CEN=0, this bit can also be cleared by hardware. It can be cleared by software by setting the ADDSENDC bit in I2C_STATC register. 0: START will not be sent 1: START will be sent
12	HEAD10R	10-bit address header executes read direction only in master receive mode 0: The 10 bit master receive address sequence is START + header of 10-bit address (write) + slave address byte 2 + RESTART + header of 10-bit address (read).

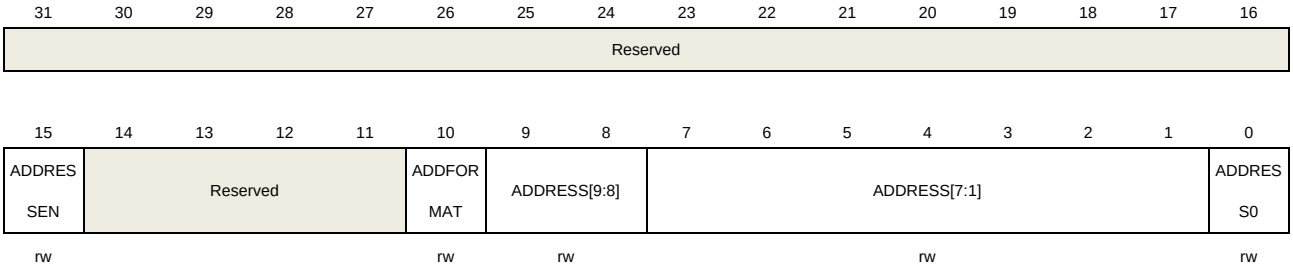
		1: The 10 bit master receive address sequence is RESTART + header of 10-bit address (read). **Note:** When the START bit is set, this bit can not be changed.
11	ADD10EN	10-bit addressing mode enable in master mode 0: 7-bit addressing in master mode 1: 10-bit addressing in master mode **Note:** When the START bit is set, this bit can not be modified.
10	TRDIR	Transfer direction in master mode 0: Master transmit 1: Master receive **Note:** When the START bit is set, this bit can not be modified.
9:0	SADDRESS[9:0]	Slave address to be sent SADDRESS[9:8]: Slave address bit 9:8 If ADD10EN = 0, these bits have no effect. If ADD10EN = 1, these bits should be written with bits 9:8 of the slave address to be sent. SADDRESS[7:1]: Slave address bit 7:1 If ADD10EN = 0, these bits should be written with the 7-bit slave address to be sent. If ADD10EN = 1, these bits should be written with bits 7:1 of the slave address to be sent. SADDRESS0: Slave address bit 0 If ADD10EN = 0, this bit has no effect. If ADD10EN = 1, this bit should be written with bit 0 of the slave address to be sent **Note:** When the START bit is set, the bit filed can not be modified.

Slave address register 0 (I2C_SADDR0)

Address offset: 0x08

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	ADDRESSEN	I2C address enable

		0: I2C address disable. 1: I2C address enable.
14:11	Reserved	Must be kept at reset value.
10	ADDFORMAT	Address mode for the I2C slave 0: 7-bit address 1: 10-bit address **Note:** When ADDRESSEN is set, this bit should not be written.
9:8	ADDRESS[9:8]	Highest two bits of a 10-bit address **Note:** When ADDRESSEN is set, this bit should not be written.
7:1	ADDRESS[7:1]	7-bit address or bits 7:1 of a 10-bit address **Note:** When ADDRESSEN is set, this bit should not be written.
0	ADDRESS0	Bit 0 of a 10-bit address **Note:** When ADDRESSEN is set, this bit should not be written.

Slave address register 1 (I2C_SADDR1)

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
ADDRESS2EN	Reserved				ADDMSK2[2:0]			ADDRESS2[7:1]							Reserved
rw					rw			rw							

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	ADDRESS2EN	Second I2C address enable 0: Second I2C address disable. 1: Second I2C address enable.
14:11	Reserved	Must be kept at reset value.
10:8	ADDMSK2[2:0]	ADDRESS2[7:1] mask Defines which bits of ADDRESS2[7:1] are compared with an incoming address byte, and which bits are masked (don't care). 000: No mask, all the bits must be compared. n(001~110): ADDRESS2[n:0] is masked. Only ADDRESS2[7:n+1] are compared.

		111: ADDRESS2[7:1] are masked. All 7-bit received addresses are acknowledged except the reserved address (0b0000xxx and 0b1111xxx). **Note:** When ADDRESS2EN is set, these bits should not be written. If ADDMSK2 is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if all the bits are matched.
7:1	ADDRESS2[7:1]	Second I2C address for the slave **Note:** When ADDRESS2EN is set, these bits should not be written.
0	Reserved	Must be kept at reset value.

Timing register (I2C_TIMING)

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
PSC[3:0]				Reserved				SCLDELY[3:0]				SDADELY[3:0]			
rw								rw				rw			

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

Bits	Fields	Descriptions
31:28	PSC[3:0]	Timing prescaler In order to generate the clock period t_{PSC} used for data setup and data hold counters, these bits are used to configure the prescaler for I2CCLK. The t_{PSC} is also used for SCL high and low level counters. $t_{PSC}=(PSC+1)*t_{I2CCLK}$
27:24	Reserved	Must be kept at reset value.
23:20	SCLDELY[3:0]	Data setup time A delay $t_{SCLDELY}$ between SDA edge and SCL rising edge can be generated by configuring these bits. And during $t_{SCLDELY}$, the SCL line is stretched low in master mode and in slave mode when SS = 0. $t_{SCLDELY}=(SCLDELY+1)*t_{PSC}$
19:16	SDADELY[3:0]	Data hold time A delay $t_{SDADELY}$ between SCL falling edge and SDA edge can be generated by configuring these bits. And during $t_{SDADELY}$, the SCL line is stretched low in master mode and in slave mode when SS = 0. $t_{SDADELY}=SDADELY*t_{PSC}$
15:8	SCLH[7:0]	SCL high period SCL high period can be generated by configuring these bits.

		$t_{SCLH}=(SCLH+1)*t_{PSC}$ **Note:** These bits can only be used in master mode.
7:0	SCLL[7:0]	SCL low period SCL low period can be generated by configuring these bits. $t_{SCLL}=(SCLL+1)*t_{PSC}$ **Note:** These bits can only be used in master mode.

Timeout register (I2C_TIMEOUT)

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
EXTOEN	Reserved			BUSTOB[11:0]											
rw				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOEN	Reserved		TOIDLE	BUSTOA[11:0]											
rw			rw	rw											

Bits	Fields	Descriptions
31	EXTOEN	Extended clock timeout detection enable When a cumulative SCL stretch time is greater than $t_{LOW:EXT}$, a timeout error will be occurred. $t_{LOW:EXT}=(BUSTOB+1)*2048*t_{I2CCLK}$. 0: Extended clock timeout detection is disabled. 1: Extended clock timeout detection is enabled.
30:28	Reserved	Must be kept at reset value.
27:16	BUSTOB[11:0]	Bus timeout B Configure the cumulative clock extension timeout. In master mode, the master cumulative clock low extend time $t_{LOW:MEXT}$ is detected. In slave mode, the slave cumulative clock low extend time $t_{LOW:SEXT}$ is detected. $t_{LOW:EXT}=(BUSTOB+1)*2048*t_{I2CCLK}$. **Note:** These bits can be modified only when EXTOEN =0.
15	TOEN	Clock timeout detection enable If the SCL stretch time greater than $t_{TIMEOUT}$ when TOIDLE =0 or high for more than t_{IDLE} when TOIDLE =1, a timeout error is detected. 0: SCL timeout detection is disabled 1: SCL timeout detection is enabled
14:13	Reserved	Must be kept at reset value.
12	TOIDLE	Idle clock timeout detection

0: BUSTOA is used to detect SCL low timeout
1: BUSTOA is used to detect both SCL and SDA high timeout when the bus is idle
Note: This bit can be written only when TOEN =0.

11:0 BUSTOA[11:0] Bus timeout A
When TOIDLE = 0, $t_{TIMEOUT}$=(BUSTOA+1)*2048*t_{I2CCLK}.
When TOIDLE = 1, t_{IDLE}=(BUSTOA+1)*4*t_{I2CCLK}.
Note: These bits can be written only when TOEN =0.

Status register (I2C_STAT)

Address offset: 0x18
Reset value: 0x0000 0001

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								READDR[6:0]							TR
								r							r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2CBSY	Reserved	SMBALT	TIMEOUT	PECERR	OUERR	LOSTARB	BERR	TCR	TC	STPDET	NACK	ADDSEND	RBNE	TI	TBE
r		r	r	r	r	r	r	r	r	r	r	r	r	rw	rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23:17	READDR[6:0]	Received match address in slave mode When the ADDSEND bit is set, these bits store the matched address. In the case of a 10-bit address, READDR[6:0] stores the header of the 10-bit address followed by the 2 MSBs of the address.
16	TR	Whether the I2C is a transmitter or a receiver in slave mode This bit is updated when the ADDSEND bit is set. 0: Receiver 1: Transmitter
15	I2CBSY	Busy flag This bit is set by hardware when a START signal is detected and cleared by hardware after a STOP signal. When I2CEN=0, this bit is also cleared by hardware. 0: No I2C communication. 1: I2C communication active.
14	Reserved	Must be kept at reset value.
13	SMBALT	SMBus Alert When SMBHAEN=1, SMBALTEN=1, and a SMBALERT event (falling edge) is detected on SMBA pin, this bit will be set by hardware. It is cleared by software by

		setting the SMBALTC bit. This bit is cleared by hardware when I2CEN=0. 0: SMBALERT event is not detected on SMBA pin 1: SMBALERT event is detected on SMBA pin
12	TIMEOUT	TIMEOUT flag. When a timeout or extended clock timeout occurred, this bit will be set. It is cleared by software by setting the TIMEOUTC bit and cleared by hardware when I2CEN=0. 0: no timeout or extended clock timeout occur 1: a timeout or extended clock timeout occur
11	PECERR	PEC error This flag is set by hardware when the received PEC does not match with the content of I2C_PEC register. Then a NACK is automatically sent. It is cleared by software by setting the PECERRC bit and cleared by hardware when I2CEN=0. 0: Received PEC and content of I2C_PEC match 1: Received PEC and content of I2C_PEC don't match, I2C will send NACK regardless of NACKEN bit.
10	OUERR	Overrun/Underrun error in slave mode In slave mode with SS=1, when an overrun/underrun error occurs, this bit will be set by hardware. It is cleared by software by setting the OUERRC bit and cleared by hardware when I2CEN=0. 0: No overrun or underrun occurs 1: Overrun or underrun occurs
9	LOSTARB	Arbitration Lost It is cleared by software by setting the LOSTARBC bit and cleared by hardware when I2CEN=0. 0: No arbitration lost. 1: Arbitration lost occurs and the I2C block changes back to slave mode.
8	BERR	Bus error When an unexpected START or STOP signal on I2C bus is detected, a bus error occurs and this bit will be set. It is cleared by software by setting BERRC bit and cleared by hardware when I2CEN=0. 0: No bus error 1: A bus error detected
7	TCR	Transfer complete reload This bit is set by hardware when RELOAD=1 and data of BYTENUM[7:0] bytes have been transferred. It is cleared by software when BYTENUM[7:0] is written to a non-zero value. 0: When RELOAD=1, transfer of BYTENUM[7:0] bytes is not completed 1: When RELOAD=1, transfer of BYTENUM[7:0] bytes is completed
6	TC	Transfer complete in master mode This bit is set by hardware when RELOAD=0, AUTOEND=0 and data of

		BYTENUM[7:0] bytes have been transferred. It is cleared by software when START bit or STOP bit is set. 0: Transfer of BYTENUM[7:0] bytes is not completed 1: Transfer of BYTENUM[7:0] bytes is completed
5	STPDET	STOP signal detected in slave mode This flag is set by hardware when a STOP signal is detected on the bus. It is cleared by software by setting STPDETC bit and cleared by hardware when I2CEN=0. 0: STOP signal is not detected. 1: STOP signal is detected.
4	NACK	Not Acknowledge flag This flag is set by hardware when a NACK is received. It is cleared by software by setting NACKC bit and cleared by hardware when I2CEN=0. 0: ACK is received. 1: NACK is received.
3	ADDSEND	Address received matches in slave mode. This bit is set by hardware when the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting ADDSENDC bit and cleared by hardware when I2CEN=0. 0: Received address not matched 1: Received address matched
2	RBNE	I2C_RDATA is not empty during receiving This bit is set by hardware when the received data is shift into the I2C_RDATA register. It is cleared when I2C_RDATA is read. 0: I2C_RDATA is empty 1: I2C_RDATA is not empty, software can read
1	TI	Transmit interrupt This bit is set by hardware when the I2C_TDATA register is empty and the I2C is ready to transmit data. It is cleared when the next data to be sent is written in the I2C_TDATA register.When SS=1, this bit can be set by software, in order to generate a TI event (interrupt if TIE=1 or DMA request if DENT =1). 0: I2C_TDATA is not empty or the I2C is not ready to transmit data 1: I2C_TDATA is empty and the I2C is ready to transmit data
0	TBE	I2C_TDATA is empty during transmitting This bit is set by hardware when the I2C_TDATA register is empty. It is cleared when the next data to be sent is written in the I2C_TDATA register. This bit can be set by software in order to empty the I2C_TDATA register. 0: I2C_TDATA is not empty 1: I2C_TDATA is empty

Status clear register (I2C_STATC)

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		SMBALTC	TIMEOUTC	PECERRC	OUERRC	LOSTARBC	BERRC	Reserved		STPDETC	NACKC	ADDSENDC	Reserved		
		w	w	w	w	w	w			w	w	w			

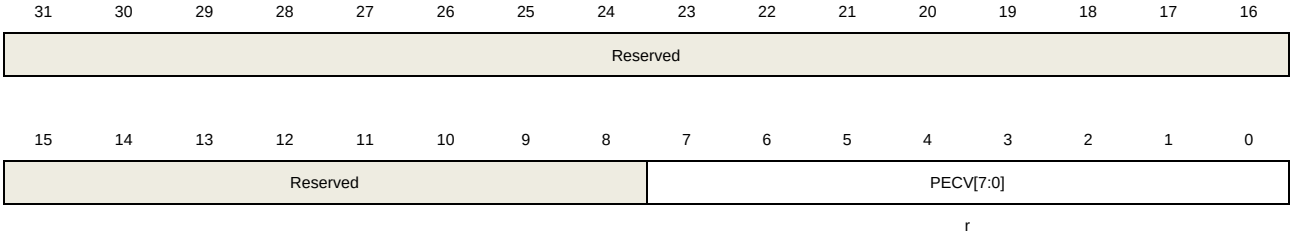
Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13	SMBALTC	SMBus alert flag clear. Software can clear the SMBALT bit of I2C_STAT by writing 1 to this bit.
12	TIMEOUTC	TIMEOUT flag clear. Software can clear the TIMEOUT bit of I2C_STAT by writing 1 to this bit.
11	PECERRC	PEC error flag clear. Software can clear the PECERR bit of I2C_STAT by writing 1 to this bit.
10	OUERRC	Overrun/Underrun flag clear. Software can clear the OUERR bit of I2C_STAT by writing 1 to this bit.
9	LOSTARBC	Arbitration Lost flag clear. Software can clear the LOSTARB bit of I2C_STAT by writing 1 to this bit
8	BERRC	Bus error flag clear. Software can clear the BERR bit of I2C_STAT by writing 1 to this bit.
7:6	Reservced	Must be kept at reset value.
5	STPDETC	STPDET flag clear Software can clear the STPDET bit of I2C_STAT by writing 1 to this bit.
4	NACKC	Not Acknowledge flag clear Software can clear the NACK bit of I2C_STAT by writing 1 to this bit.
3	ADDSENDC	ADDSEND flag clear Software can clear the ADDSEND bit of I2C_STAT by writing 1 to this bit.
2:0	Reserved	Must be kept at reset value.

PEC register (I2C_PEC)

Address offset: 0x20

Reset value: 0x0000 0000

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



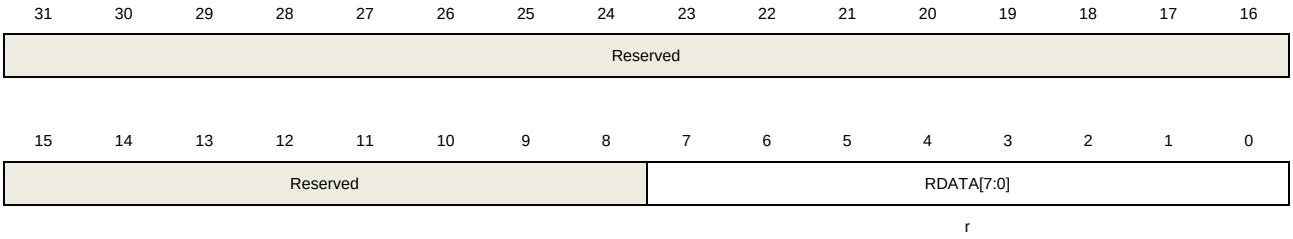
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	PECV[7:0]	Packet Error Checking Value that calculated by hardware when PEC is enabled. PECV is cleared by hardware when I2CEN = 0.

Receive data register (I2C_RDATA)

Address offset: 0x24

Reset value: 0x0000 0000

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



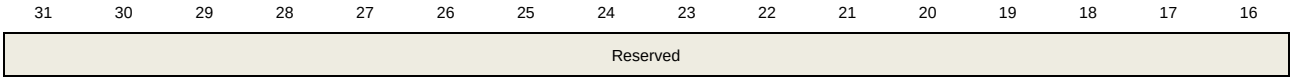
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	RDATA[7:0]	Receive data value

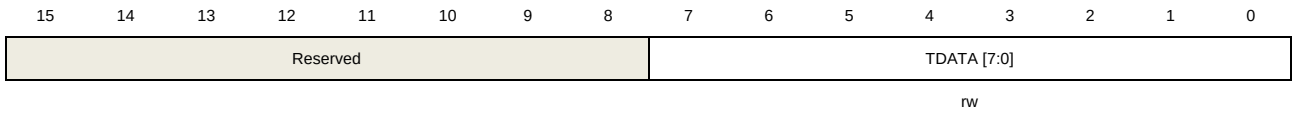
Transmit data register (I2C_TDATA)

Address offset: 0x28

Reset value: 0x0000 0000

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





Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	TDATA[7:0]	Transmit data value

22. Serial peripheral interface/Inter-IC sound (SPI/I2S)

22.1. 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. Quad-SPI master mode is also supported in SPI0.

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. (By using two extra I2S modules called I2S1_ADD and I2S2_ADD, I2S full duplex mode is also supported in SPI1 and SPI2.)

22.2. Characteristics

22.2.1. SPI characteristics

- Master or slave operation with full-duplex or half-duplex or simplex mode.
- Separate transmission and reception buffer, 16 bits wide.
- Data frame size can be 8 or 16 bits.
- Bit order can be LSB or MSB.
- Software and hardware NSS management.
- Hardware CRC calculation, transmission and checking.
- Transmission and reception using DMA.
- SPI TI mode supported.
- SPI NSS pulse mode supported.
- Quad-SPI configuration available in master mode (only in SPI0).

22.2.2. I2S characteristics

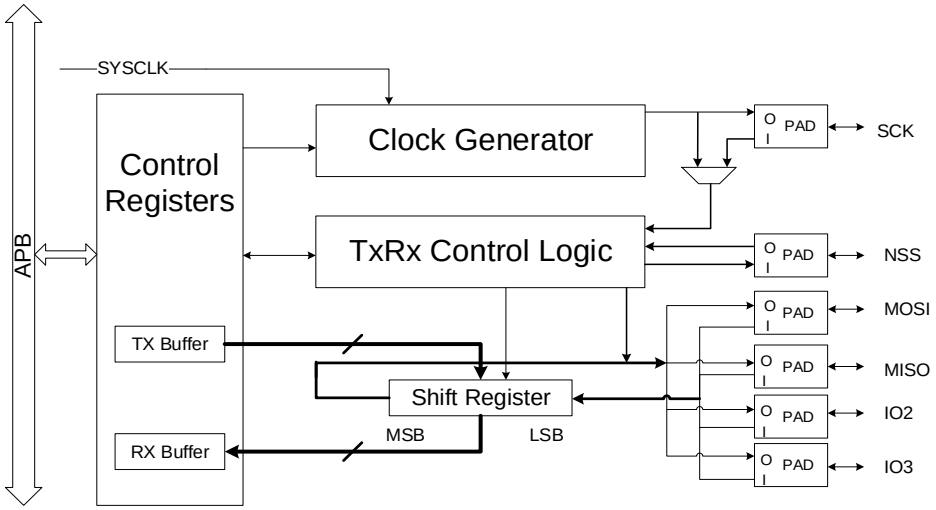
- Master or slave operation for transmission/reception.
- Master or slave operation with full-duplex mode(only in SPI1 and SPI2)
- Four I2S standards supported: Phillips, MSB justified, LSB justified and PCM standard.
- Data length can be 16 bits, 24 bits or 32 bits.
- Channel length can be 16 bits or 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.

22.3. SPI function overview

22.3.1. SPI block diagram

Figure 22-1. Block diagram of SPI



22.3.2. SPI signal description

Normal configuration (not Quad-SPI mode)

Table 22-1. 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

Pin name	Direction	Description
		Master in hardware NSS mode: when NSSDRV=1, it is NSS output, suitable for single master application; when NSSDRV=0, it is NSS input, suitable for multi-master application. Slave in hardware NSS mode: NSS input, as a chip select signal for slave.

Quad-SPI configuration

SPI is in single wire mode by default and enters into Quad-SPI mode after QMOD bit in SPI_QCTL register is set (only available in SPI0). Quad-SPI mode can only work in master mode.

The IO2 and IO3 pins can be driven high in normal Non-Quad-SPI mode by configuring IO23_DRV bit in SPI_QCTL register.

The SPI is connected to external devices through 6 pins in Quad-SPI mode:

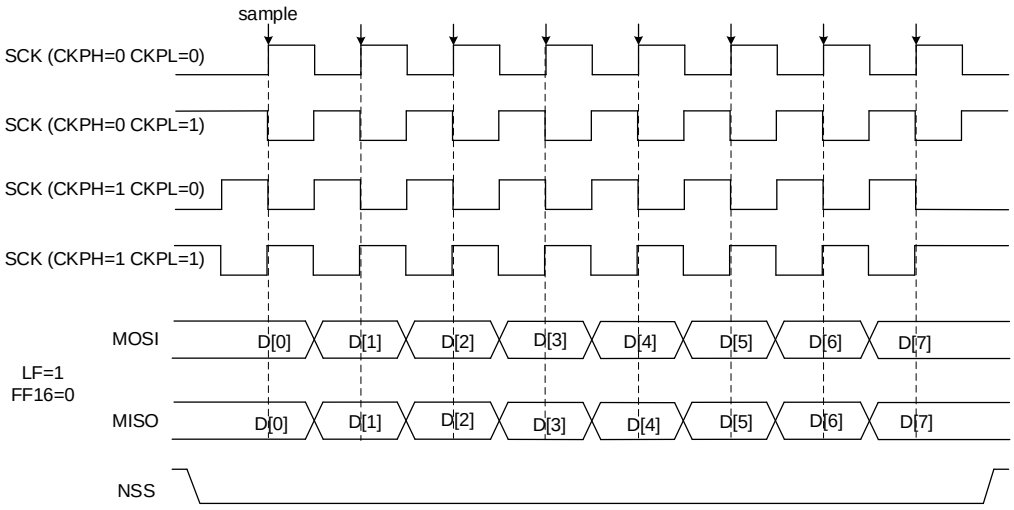
Table 22-2. Quad-SPI signal description

Pin name	Direction	Description
SCK	O	SPI clock output
MOSI	I/O	Transmission/Reception data 0
MISO	I/O	Transmission/Reception data 1
IO2	I/O	Transmission/Reception data 2
IO3	I/O	Transmission/Reception data 3
NSS	O	NSS output

22.3.3. 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 SPI is in idle state and CKPH bit decides either first or second clock edge is a valid sampling edge. These bits take no effect in TI mode.

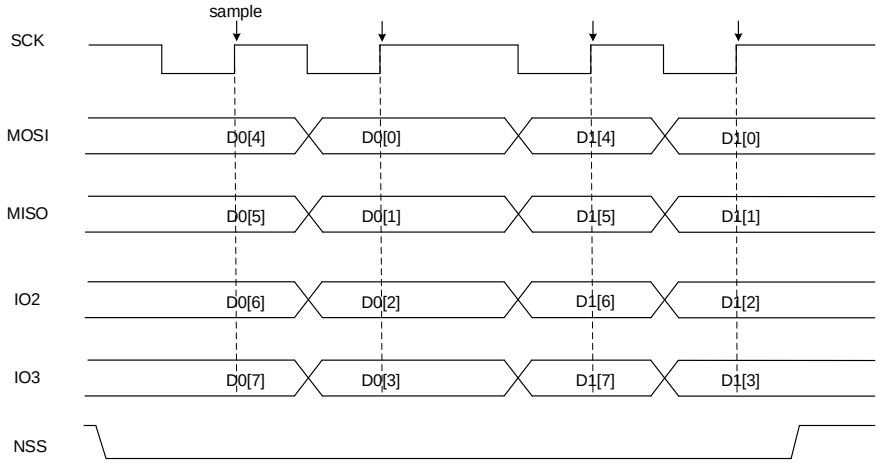
Figure 22-2. SPI timing diagram in normal mode



In SPI 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 send the LSB first if LF=1, or the MSB if LF=0. The data order is fixed to MSB first in TI mode.

Figure 22-3. SPI timing diagram in Quad-SPI mode (CKPL=1, CKPH=1, LF=0)



The data frame length is fixed to 8 bits in Quad-SPI mode.

Data order is configured by LF bit in SPI_CTL0 register, and SPI will send the LSB first if LF=1, or the MSB if LF=0. The data order is fixed to MSB first in TI mode.

When the SPI_DATA register is accessed, data frames are always right-aligned into either a byte (If the data length is less than or equal to one byte) or a half-word. During communication, only bits within the data frame will be output with the clock.

22.3.4. 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.

Table 22-3. NSS function in slave mode

Mode	Register configuration	Description
Slave hardware NSS mode	MSTMOD = 0 SWNSSEN = 0	SPI slave gets NSS level from NSS pin.
Slave software NSS mode	MSTMOD = 0 SWNSSEN = 1	SPI slave NSS level is determined by the SWNSS bit. SWNSS = 0: NSS level is low SWNSS = 1: NSS level is high

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 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 goes low after SPI is enabled.

The application may also use a general purpose IO as NSS pin to realize more flexible NSS.

Table 22-4. NSS function in master mode

Mode	Register configuration	Description
Master hardware NSS output mode	MSTMOD = 1 SWNSSEN = 0 NSSDRV=1	Applicable to single-master mode. The master uses the NSS pin to control the SPI slave device. At this time, the NSS is configured as the hardware output mode. NSS goes low after enabling SPI.
Master hardware NSS input mode	MSTMOD = 1 SWNSSEN = 0 NSSDRV=0	Applicable to multi-master mode. At this time, NSS is configured as hardware input mode. Once the NSS pin is pulled low, SPI will automatically enter slave mode, and a master

Mode	Register configuration	Description
		configuration error will occur and the CONFERR bit will be set to 1.
Master software NSS mode	MSTMOD = 1 SWNSSEN = 1 SWNSS = 0 NSSDRV: Don't care	Applicable to multi-master mode. Once SWNSS = 0, SPI will automatically enter slave mode, and a master configuration error will occur and the CONFERR bit will be 1.
	MSTMOD = 1 SWNSSEN = 1 SWNSS = 1 NSSDRV: Don't care	The slave can use hardware or software NSS mode.

22.3.5. SPI operation modes

Table 22-5. 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	MOSI: not used MISO: transmission

Mode	Description	Register configuration	Data pin usage
		BDEN = 0 BDOEN: Don't care	
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 bidirectional connection	MSTMOD = 0 RO = 0 BDEN = 1 BDOEN = 1	MOSI: not used MISO: transmission
SRB	Slave reception with bidirectional connection	MSTMOD = 0 RO = 0 BDEN = 1 BDOEN = 0	MOSI: not used MISO: reception

Figure 22-4. A typical full-duplex connection

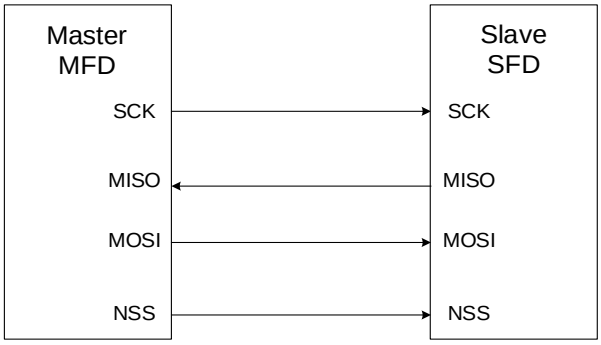


Figure 22-5. A typical simplex connection (Master: receive, Slave: transmit)

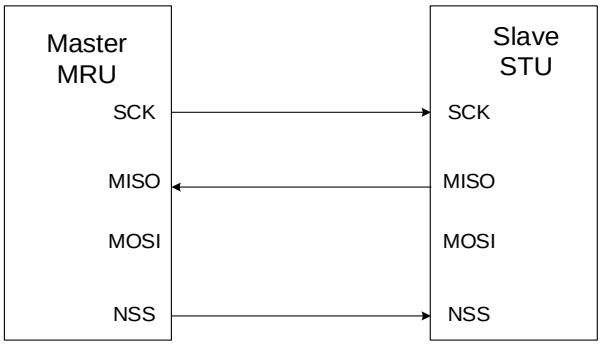


Figure 22-6. A typical simplex connection (Master: transmit only, Slave: receive)

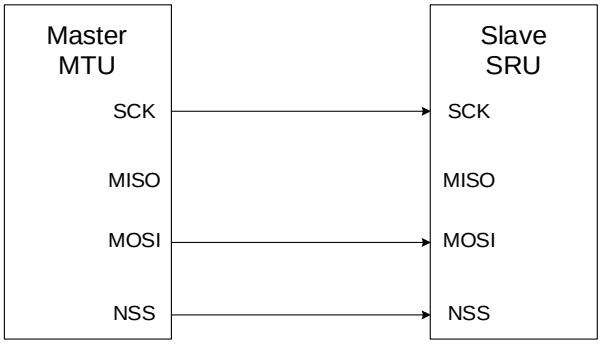
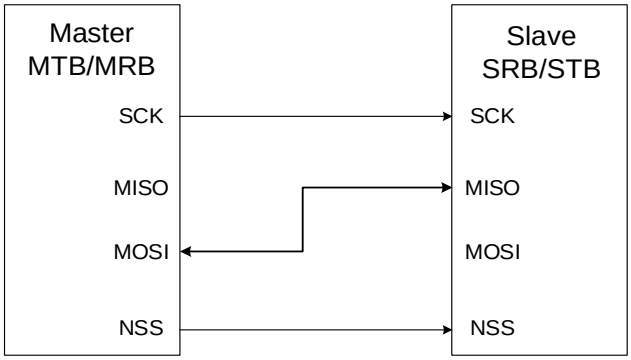


Figure 22-7. A typical bidirectional connection



Initialization sequence

Before transmitting or receiving data, application should follow the SPI initialization sequence described below:

1. If master mode or slave TI mode is used, program the PSC [2:0] bits in SPI_CTL0 register to generate SCK with desired baud rate or configure the Td time in TI mode, otherwise, ignore this step.
2. Configure data format (FF16 bit in the SPI_CTL0 register).
3. Configure the clock timing register (CKPL and CKPH bits in the SPI_CTL0 register).
4. Configure the frame format (LF bit in the SPI_CTL0 register).
5. Configure 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. If TI mode is used, set TMOD bit in SPI_CTL1 register, otherwise, ignore this step.
7. If NSSP mode is used, set NSSP bit in SPI_CTL1 register, otherwise, ignore this step.
8. Configure MSTMOD, RO, BDEN and BDOEN depending on the operating modes described in ***SPI operation modes*** section.
9. Enable the SPI (set the SPIEN bit).

Note: During communication, CKPH, CKPL, MSTMOD, PSC[2:0] and LF bits should not be changed.

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 transmission 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 transmission buffer before the transmission starts in slave mode.

When SPI begins to send a data frame, it first loads this data frame from the transmission buffer to the shift register and then begins to transmit the loaded data frame. After TBE flag is set, which means the transmission 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

After the last valid sample clock, the incoming data will be moved from shift register to the reception buffer and RBNE will be set. The application should read SPI_DATA register to get the received data and this will clear the RBNE flag automatically when reception buffer is empty. 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 transmission buffer is not empty.

SPI operation sequence in different modes (not Quad-SPI, TI mode or NSSP mode)

In full-duplex mode, either MFD or SFD, the RBNE and TBE flags should be monitored and then follow the sequences described above.

The transmission mode (MTU, MTB, STU or STB) is similar to the transmission sequence of full-duplex mode except that the RBNE bit and RXORERR bit need to be ignored.

The master reception mode (MRU or MRB) is different from the reception sequence of full-duplex mode. In MRU or MRB mode, after SPI is enabled, the SPI continuously generates SCK 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 the reception sequence of full-duplex mode except that the TBE bit need to be ignored.

SPI TI mode

SPI TI mode takes NSS as a special frame header flag signal and its operation sequence is

similar to normal mode described above. The modes described above (MFD, MTU, MRU, MTB, MRB, SFD, STU, SRU, STB and SRB) are still supported in TI mode. While, in TI mode the CKPL and CKPH bits in SPI_CTL0 registers take no effect and the SCK sample edge is falling edge.

Figure 22-8. Timing diagram of TI master mode with discontinuous transfer

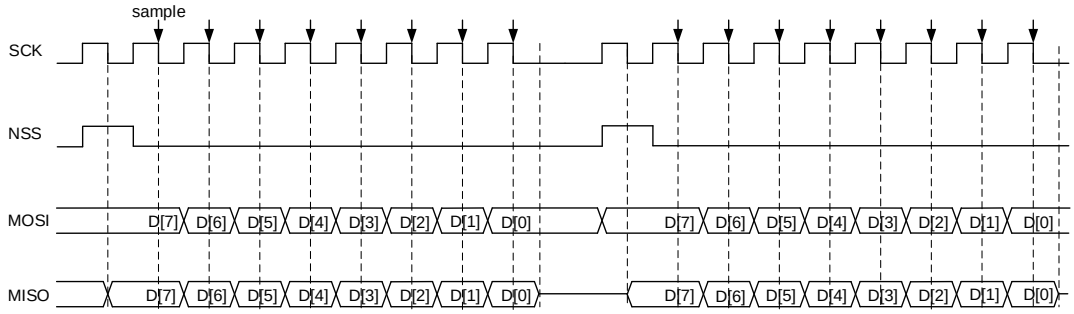
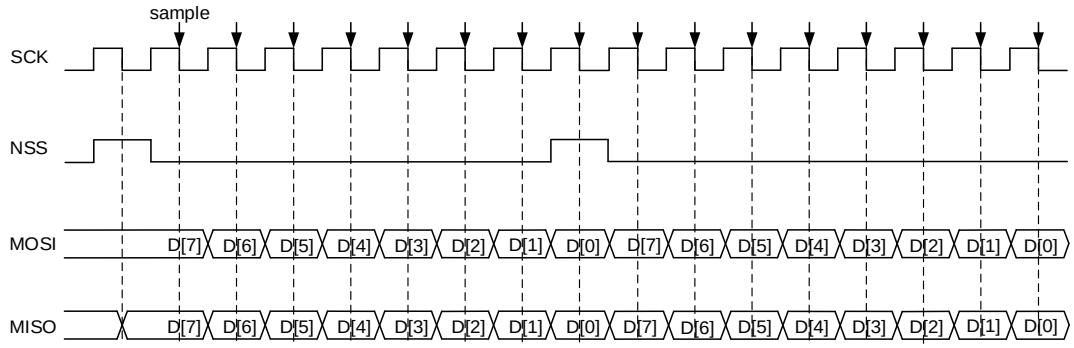
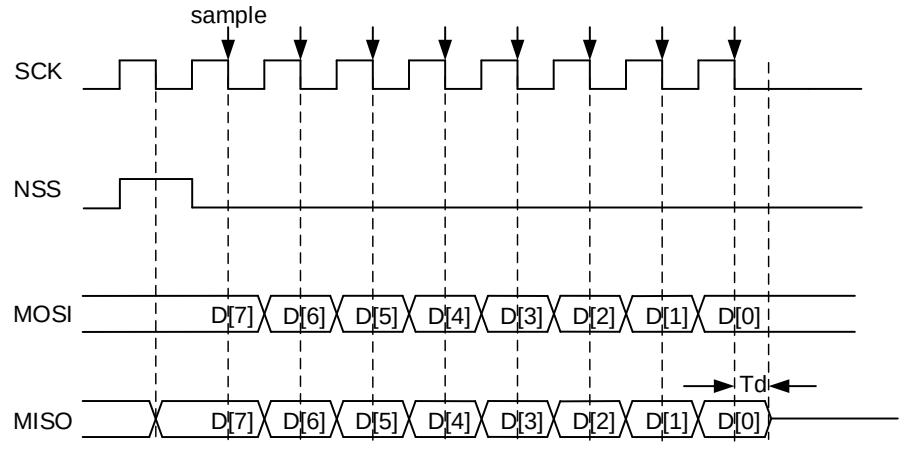


Figure 22-9. Timing diagram of TI master mode with continuous transfer



In master TI mode, SPI can perform continuous or non-continuous transfer. If the master writes SPI_DATA register fast enough, the transfer is continuous, otherwise non-continuous. In non-continuous transfer there is an extra header clock cycle before each byte. While in continuous transfer, the extra header clock cycle only exists before the first byte and the following bytes' header clock is overlaid at the last bit of pervious bytes.

Figure 22-10. Timing diagram of TI slave mode



In slave TI mode, after the last rising edge of SCK in transfer, the slave begins to transmit the

LSB bit of the last data byte, and after a half-bit time, the master begins to sample the line. To make sure that the master samples the right value, the slave should continue to drive this bit after the falling sample edge of SCK for a period of time before releasing the pin. This time is calledT_d, T_d is decided by PSC [2:0] bits in SPI_CTL0 register.

$$T_d=\frac{T_{bit}}{2}+5*T_{pclk} \quad (22\text{-}1)$$

For example, if PSC [2:0] = 010, T_d is 9*T_{pclk}.

In slave mode, the slave also monitors the NSS signal and sets an error flag FERR if it detects an incorrect NSS behavior, for example: toggles at the middle bit of a byte.

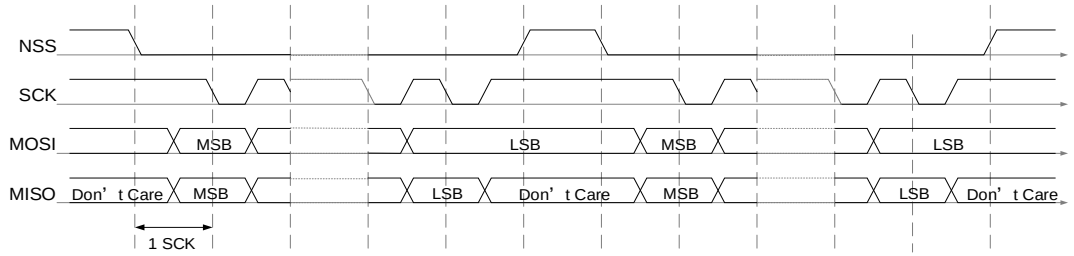
NSS pulse mode operation sequence

This function is controlled by NSSP bit in SPI_CTL1 register. In order to implement this function, several additional conditions must be met: configure the device to master mode, frame format should follow the normal SPI protocol, select the first clock transition as the data capture edge.

In summary, MSTMOD = 1, NSSP = 1, CKPH = 0.

When NSS pulse mode is enabled, a pulse duration of at least 1 SCK clock period is inserted between two successive data frames depending on the status of internal data transmission buffer. Multiple SCK clock cycle intervals are possible if the transfer buffer stays empty. This function is designed for single master-slave configuration for the slave to latch data. The following diagram depicts its timing diagram.

Figure 22-11. Timing diagram of NSS pulse with continuous transmit



Quad-SPI mode operation sequence

The Quad-SPI mode is designed to control Quad-SPI Flash.

In order to enter Quad-SPI mode, the software should first verify that the TBE bit is set and TRANS bit is cleared, then set QMOD bit in SPI_QCTL register. In Quad-SPI mode, BDEN, BDOEN, CRCEN, CRCNT, CRCL, RO and LF in SPI_CTL0 register should be kept cleared and DZ[3:0] should be set to ensure that SPI data size is 8-bit, MSTMOD should be set to ensure that SPI is in master mode. SPIEN, PSC, CKPL and CKPH should be configured as desired.

There are two operation modes in Quad-SPI mode: quad write and quad read, decided by QRD bit in SPI_QCTL register.

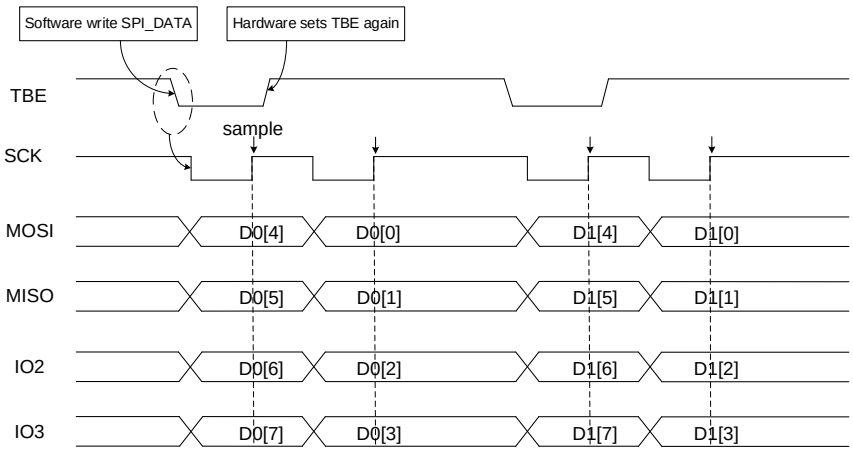
Quad write operation

SPI works in quad write mode when QMOD is set and QRD is cleared in SPI_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as output pins. SPI begins to generate clock on SCK line and transmit data on MOSI, MISO, IO2 and IO3 as soon as data is written into SPI_DATA (TBE is cleared) and SPIEN is set. Once SPI starts transmission, it always checks TBE status at the end of a frame and stops when condition is not met.

The operation flow for transmitting in quad mode:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI_CTL0 and SPI_CTL1 based on application requirements.
2. Set QMOD bit in SPI_QCTL register and then enable SPI by setting SPIEN in SPI_CTL0.
3. Write a byte to SPI_DATA register and the TBE will be cleared.
4. Wait until TBE is set by hardware again before writing the next byte.

Figure 22-12. Timing diagram of write operation in Quad-SPI mode



Quad read operation

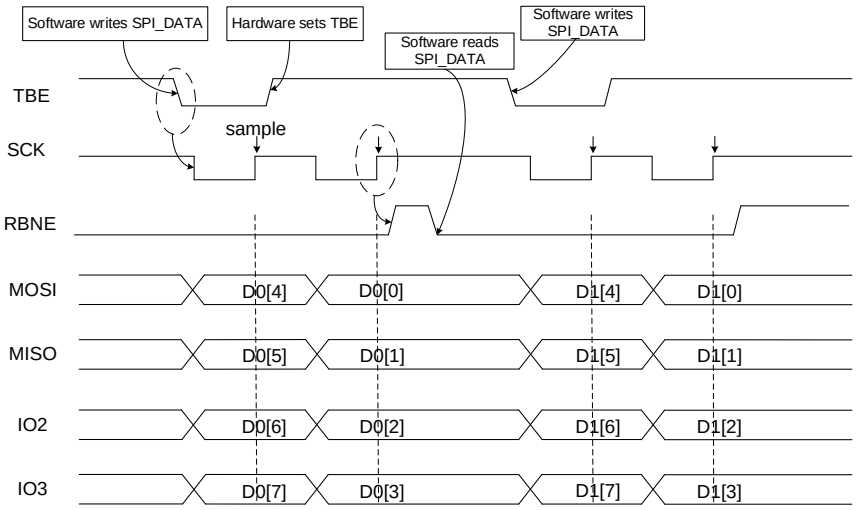
SPI works in quad read mode when QMOD and QRD are both set in SPI_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as input pins. SPI begins to generate clock on SCK line as soon as a data is written into SPI_DATA (TBE is cleared) and SPIEN is set. Writing data into SPI_DATA is only to generate SCK clocks, so the written data can be any value. Once SPI starts transmission, it always checks SPIEN and TBE status at the end of a frame and stops when condition is not met. So, dummy data should always be written into SPI_DATA to generate SCK.

The operation flow for receiving in quad mode is shown below:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI_CTL0 and SPI_CTL1 register based on application requirements.
2. Set QMOD and QRD bits in SPI_QCTL register and then enable SPI by setting SPIEN in SPI_CTL0 register.
3. Write an arbitrary byte (for example, 0xFF) to SPI_DATA register.

4. Wait until the RBNE flag is set and read SPI_DATA to get the received byte.
5. Write an arbitrary byte (for example, 0xFF) to SPI_DATA to receive the next byte.

Figure 22-13. Timing diagram of read operation in Quad-SPI mode



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 ongoing transfer completed.

TI mode

The disabling sequence of TI mode is the same as the sequences described above.

NSS pulse mode

The disabling sequence of NSSP mode is the same as the sequences described above.

Quad-SPI mode

Before leaving quad wire mode or disabling SPI, software should first check that TBE bit is set and TRANS bit is cleared, then the QMOD bit in SPI_QCTL register and SPIEN bit in SPI_CTL0 register are cleared.

22.3.6. DMA function

The DMA 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 configure DMA modules correctly, 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 when 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 when RBNE = 1, then DMA will acknowledge to this request and read data from the SPI_DATA register automatically.

22.3.7. CRC function

There are two CRC calculators in SPI: one for transmission and the other for reception. The CRC calculation uses the polynomial defined in SPI_CRCPOLY register.

Application can enable the CRC function by setting CRCEN bit in SPI_CTL0 register. The CRC calculators calculate CRC for each bit transmitted and received on lines continuously, and the calculated CRC values can be read from SPI_TCRC and SPI_RCRC registers.

To transmit the calculated CRC value, application should set the CRCNT bit in SPI_CTL0 register after the last data is written to the transmission buffer. In full-duplex mode (MFD or SFD), when the SPI transmits a CRC and prepares to check the received CRC value, the SPI treats the incoming data as a 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 the DMA function is enabled, the software does not need to set the CRCNT bit, and the hardware will handle the CRC transmission and verification automatically.

Note: When SPI is in slave mode and CRC function is enable, the CRC calculator is sensitive to input SCK clock whether SPI is enable or not. The software must enable CRC only when the clock is stable to avoid wrong CRC calculation. And when SPI works as a slave, the NSS internal signal needs to be kept low between the data phase and CRC phase.

22.3.8. SPI interrupts

Status flags

- **Transmission buffer empty flag (TBE)**

This bit is set when the transmission buffer is empty, the software can write the next data to the transmission buffer by writing the SPI_DATA register.

- **Reception buffer not empty flag (RBNE)**

This bit is set when reception buffer is not empty, which means that one data is received and stored in the reception buffer, and software can read the data by reading the SPI_DATA register.

- **SPI transmitting ongoing flag (TRANS)**

TRANS is a status flag to indicate whether the transfer is ongoing or not. It is set and cleared by hardware and not controlled by software. This flag doesn't generate any interrupt.

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. For The reception buffer contents won't be covered with the newly incoming data, so the newly incoming data is lost.

- **Format error (FERR)**

In slave TI mode, the slave also monitors the NSS signal and set an error flag if it detects an incorrect NSS behavior, for example: toggles at the middle bit of a byte.

- **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.

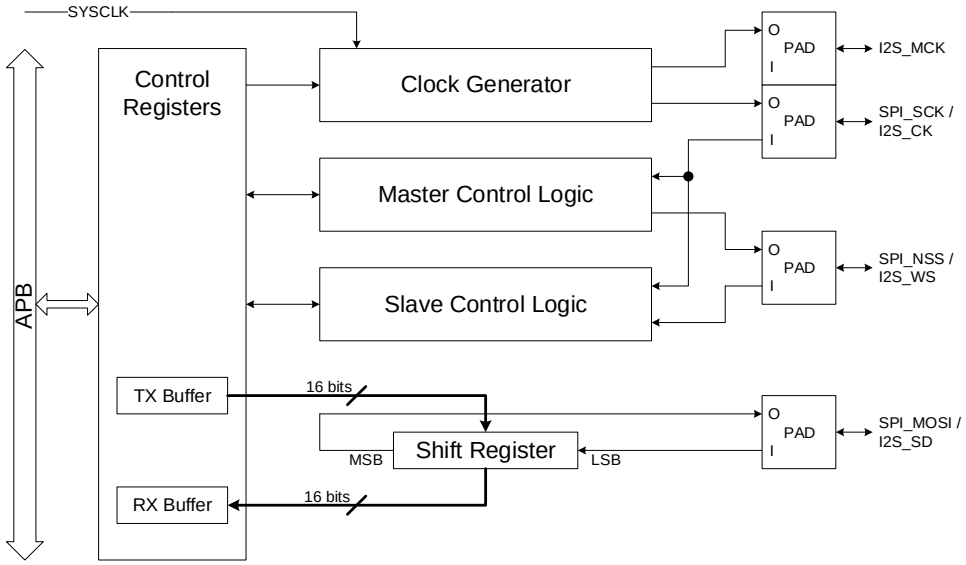
Table 22-6. SPI interrupt requests

Flag	Description	Clear method	Interrupt enable bit
TBE	Transmission buffer empty	Write SPI_DATA register.	TBEIE
RBNE	Reception 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	
FERR	TI mode format error	Write 0 to FERR bit	

22.4. I2S function overview

22.4.1. I2S block diagram

Figure 22-14. 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.

22.4.2. 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.

22.4.3. 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 bit 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 the transmission of a frame. In the case that the data length is 16-bit, only one write or read operation to or from the SPI_DATA register is needed to complete the transmission of 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, and I2S_WS becomes valid one clock before the data. The timing diagrams for each configuration are shown below.

Figure 22-15. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

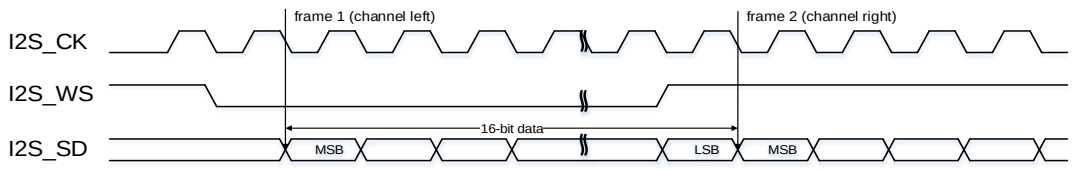
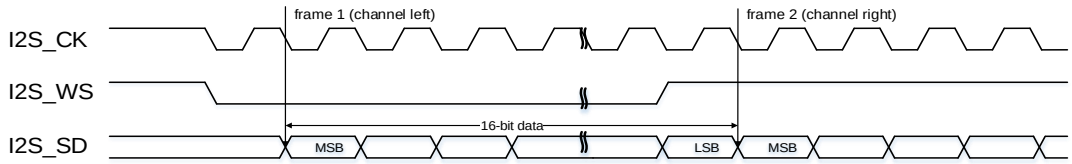


Figure 22-16. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)



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 the transmission of a frame.

Figure 22-17. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

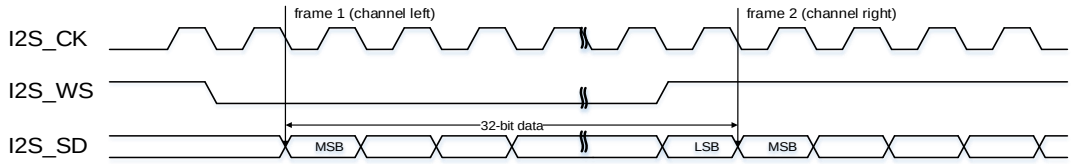
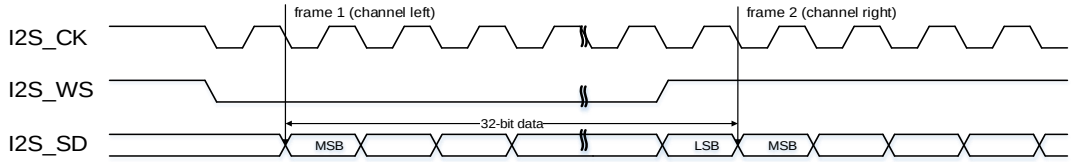


Figure 22-18. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)



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 the transmission of 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 22-19. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

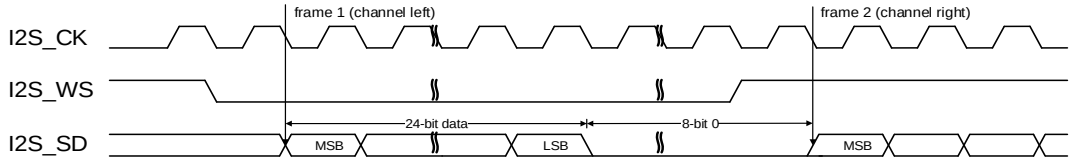
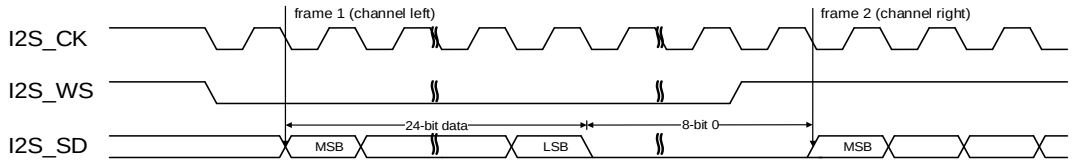


Figure 22-20. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)



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 22-21. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

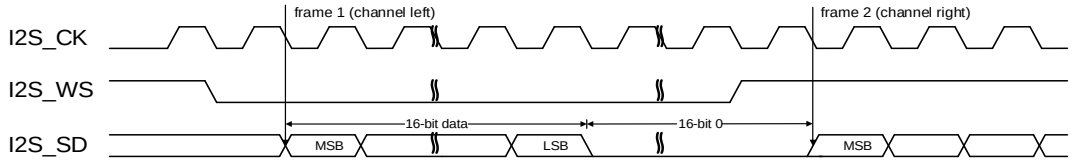
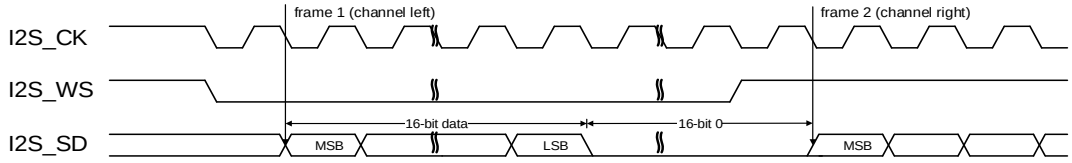


Figure 22-22. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



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 the transmission of a frame. The remaining 16 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 22-23. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

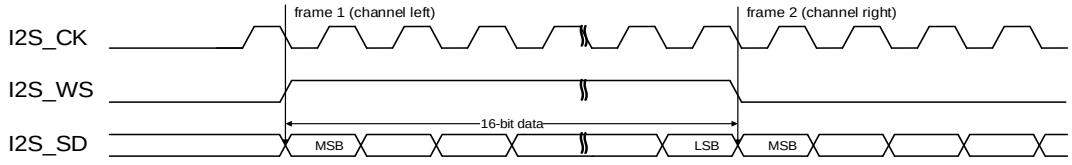


Figure 22-24. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)

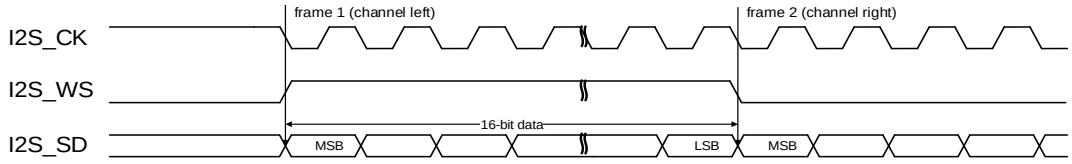


Figure 22-25. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

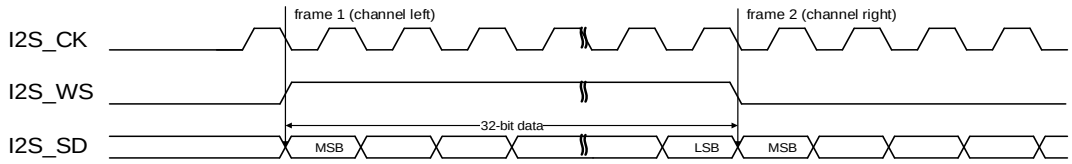


Figure 22-26. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)

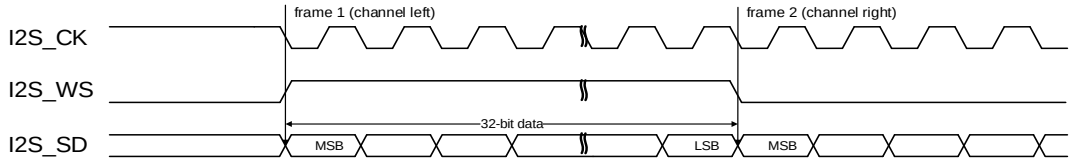


Figure 22-27. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

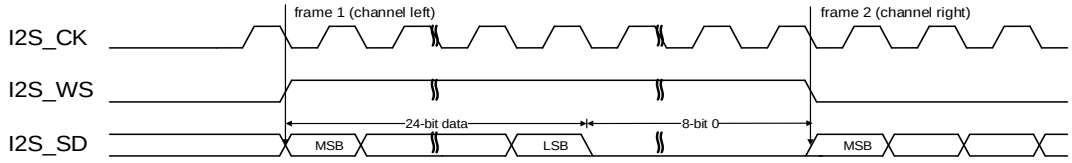


Figure 22-28. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

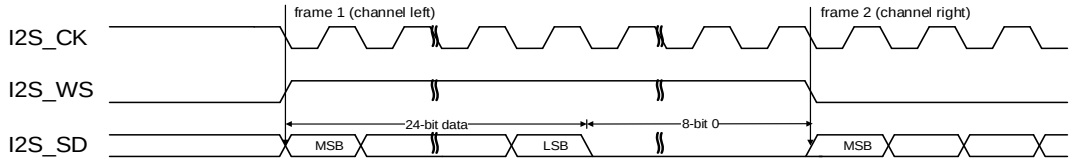


Figure 22-29. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

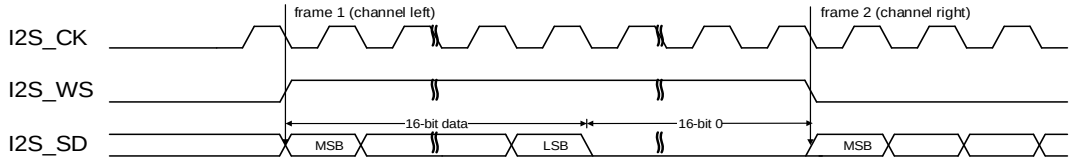
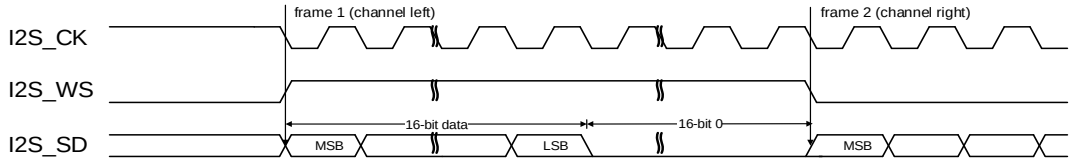


Figure 22-30. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



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 22-31. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

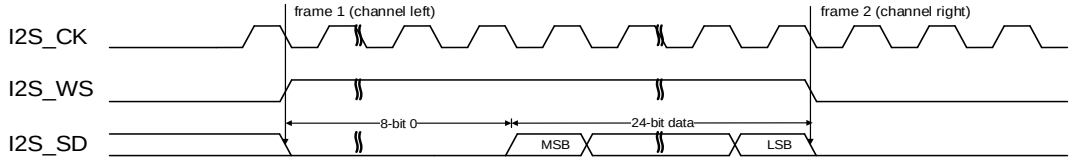
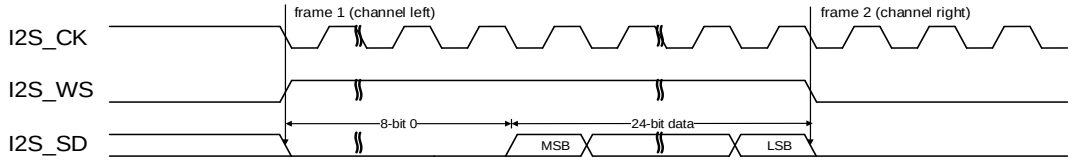


Figure 22-32. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)



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 the transmission of 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 22-33. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

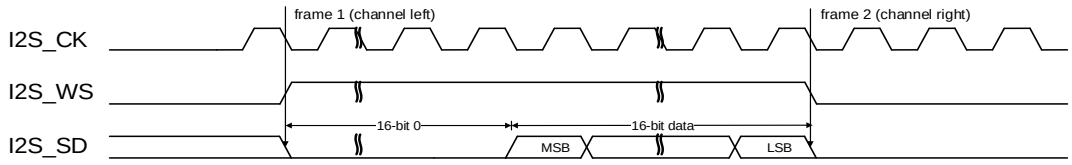
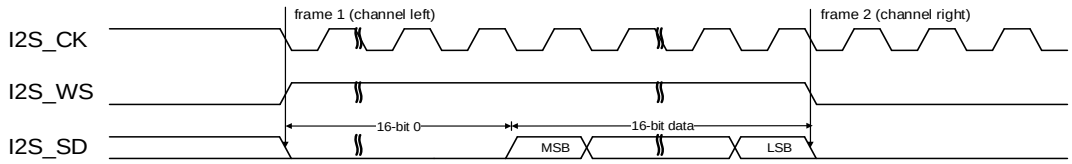


Figure 22-34. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



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 the transmission of a frame. The remaining 16 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 22-35. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

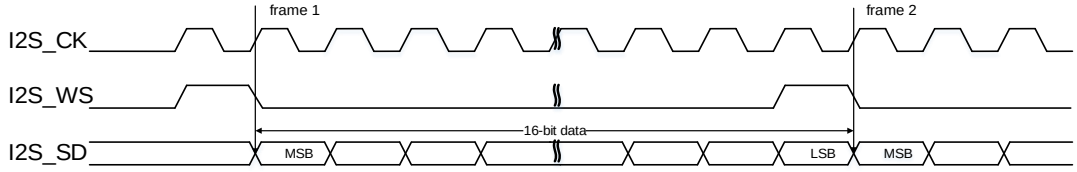


Figure 22-36. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)

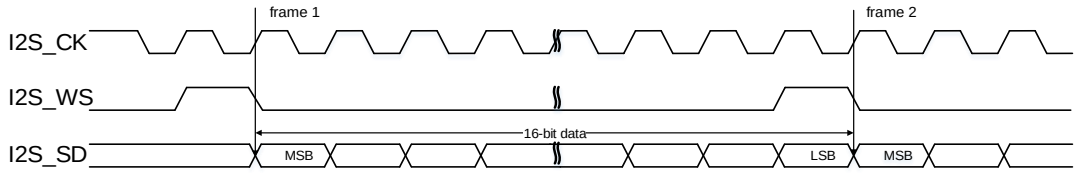


Figure 22-37. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

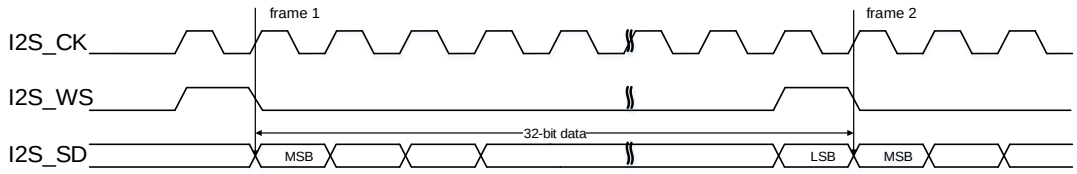


Figure 22-38. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)

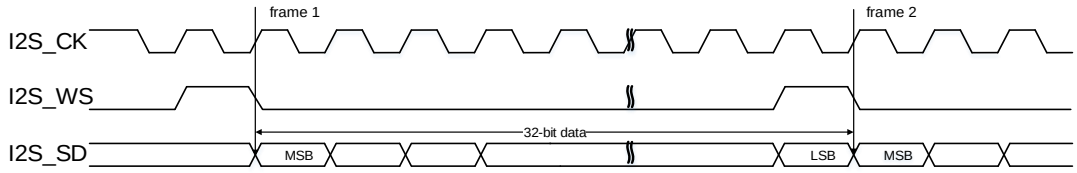


Figure 22-39. PCM standard short frame synchronization mode timing diagram

(DTLEN=01, CHLEN=1, CKPL=0)

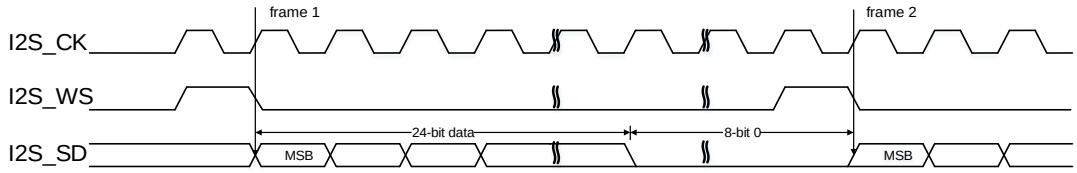


Figure22-40. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

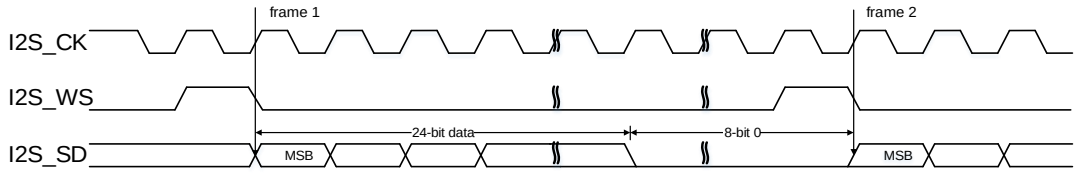


Figure 22-41. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

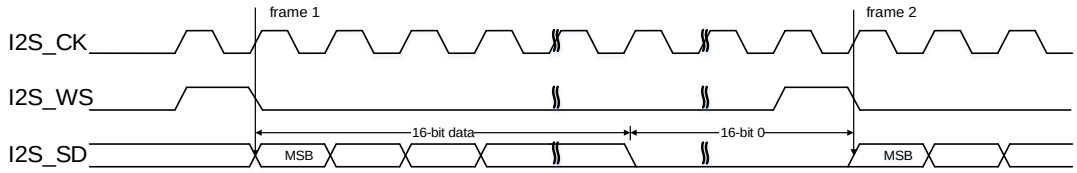
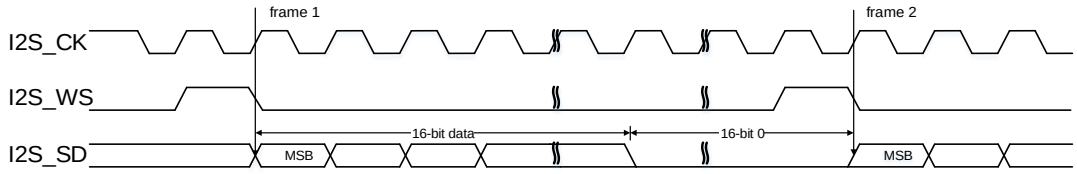


Figure 22-42. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



The timing diagrams for each configuration of the long frame synchronization mode are shown below.

Figure 22-43. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

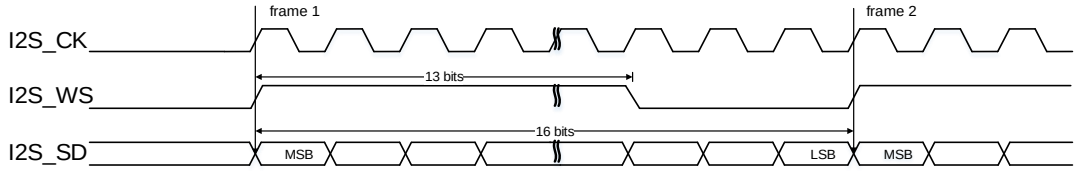


Figure22-44. PCM standard long frame synchronization mode timing diagram

(DTLEN=00, CHLEN=0, CKPL=1)

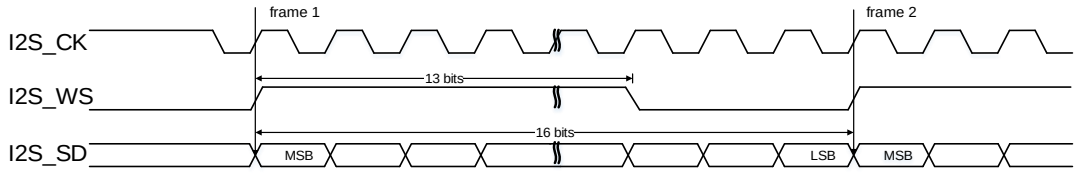


Figure 22-45. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

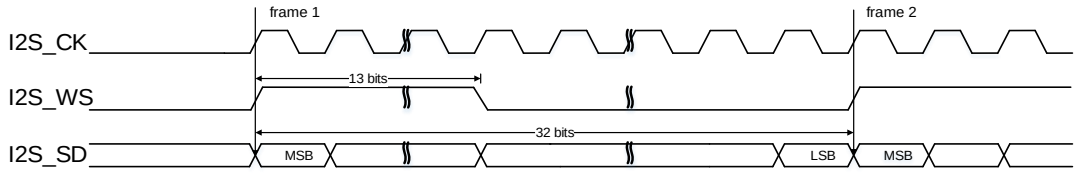


Figure 22-46. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)

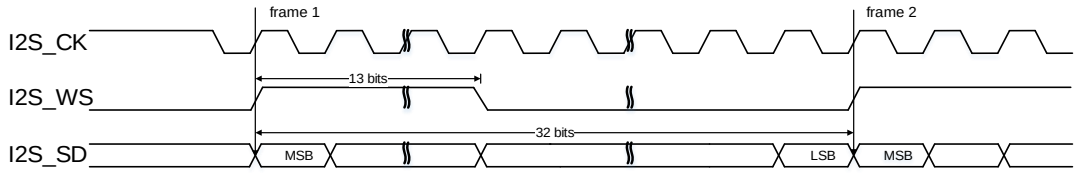


Figure 22-47. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

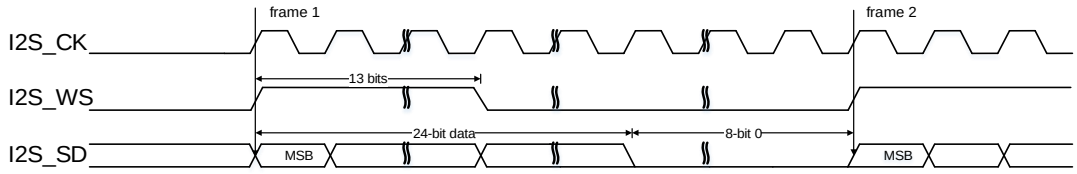


Figure 22-48. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

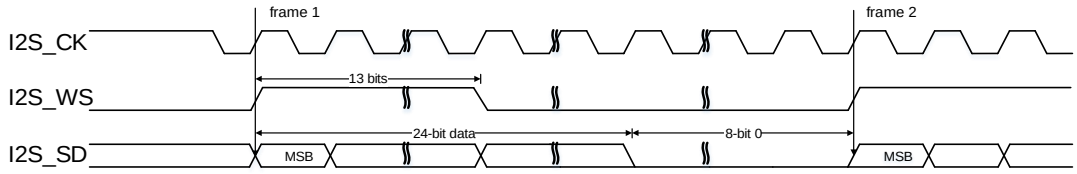


Figure 22-49. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

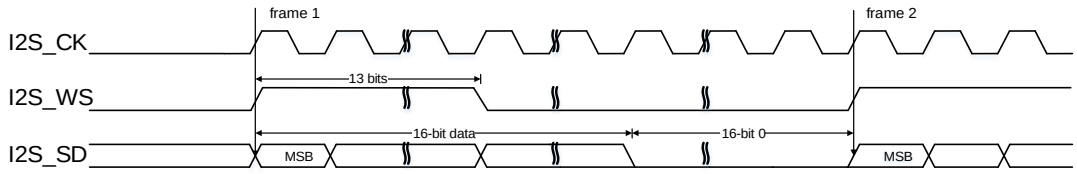
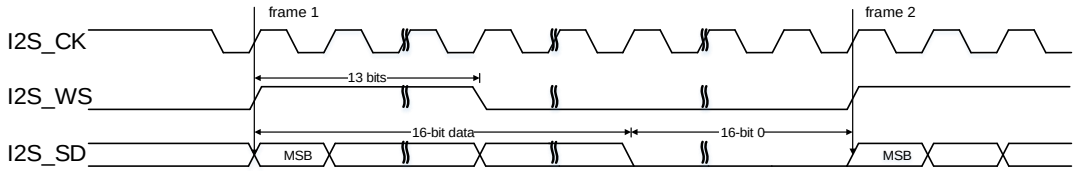
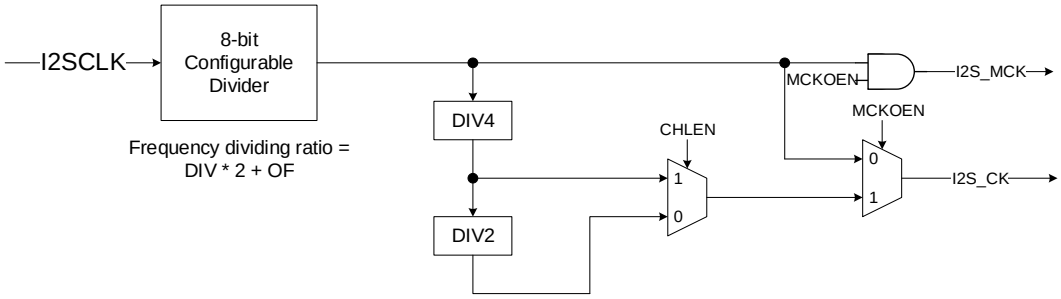


Figure 22-50. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



22.4.4. I2S clock

Figure 22-51. Block diagram of I2S clock generator



The block diagram of I2S clock generator is shown as ***Figure 22-51. 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 clock is the system clock(CK_SYS) or PLL2*2. The I2S bitrate can be calculated by the formulas shown in ***Table 22-7. I2S bitrate calculation formulas***.

Table 22-7. 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 22-8. Audio sampling frequency calculation formulas***.

Table 22-8. 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))

22.4.5. Operation

Operation modes

The operation mode is selected by the I2SOPMOD[1:0] 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 22-9. Direction of I2S interface signals for each operation mode***.

Table 22-9. Direction of I2S interface signals for each operation mode

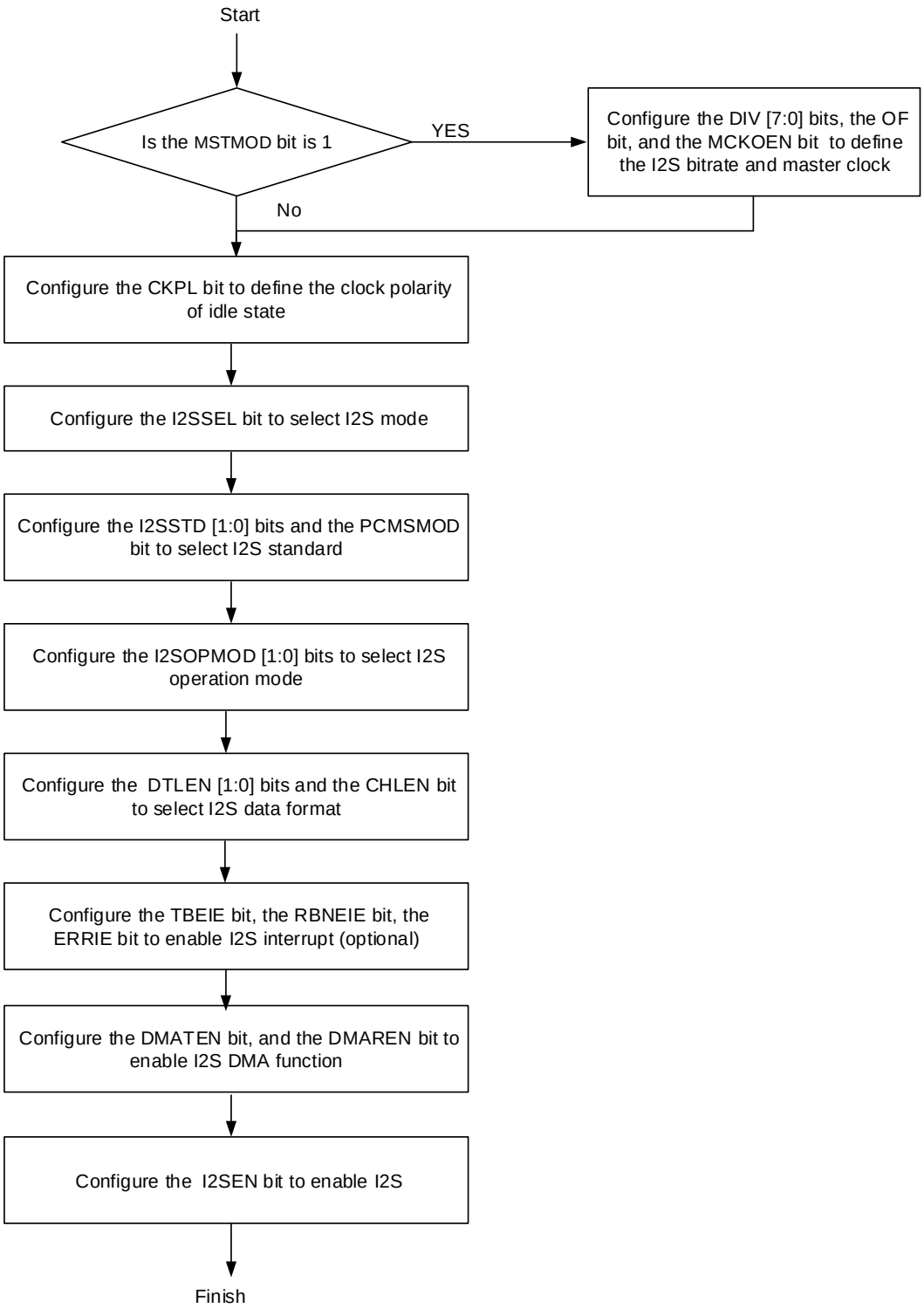
Operation mode	I2S_MCK	I2S_CK	I2S_WS	I2S_SD	I2S_ADD_SD(2)
Master transmission	Output or NU(1)	Output	Output	Output	NU(1)
Master reception	Output or NU(1)	Output	Output	Input	NU(1)
Slave transmission	Input or NU(1)	Input	Input	Output	NU(1)
Slave reception	Input or NU(1)	Input	Input	Input	NU(1)
Full-duplex	output or NU(1)	output	output	output or input	Input or output

1. NU means the pin is not used by I2S and can be used by other functions.
2. To support full-duplex operation in I2S1 and I2S2, there are two extra I2S modules in chip called I2S_ADD1 and I2S_ADD2. I2S_ADD_SD is the data pin of I2S_ADD module. Full-duplex will be described later in this chapter.

I2S initialization sequence

I2S initialization sequence is shown as below ***Figure 22-52. I2S initialization sequence***.

Figure 22-52. I2S initialization sequence



I2S master transmission sequence

The TBE flag is used to control the transmission sequence. As is mentioned before, the TBE flag indicates that the transmission buffer is empty, and an interrupt will be generated if the TBEIE bit in the SPI_CTL1 register is set. At the beginning, the transmission 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 transmission 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 transmission 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 which channel side the data to transfer belongs to. 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 disable I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

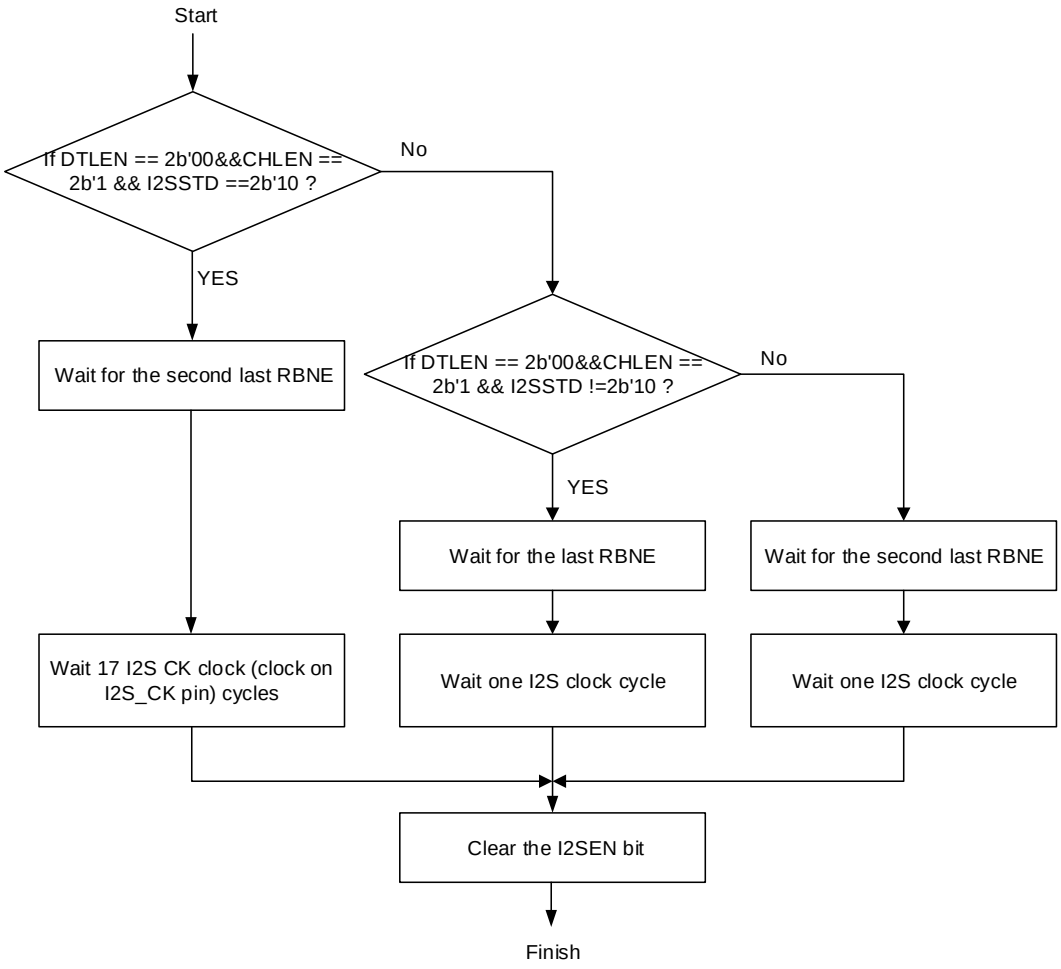
I2S master reception sequence

The RBNE flag is used to control the reception sequence. As is mentioned before, the RBNE flag indicates the reception 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 reception buffer is empty (RBNE is low). When a reception sequence finishes, the received data in the shift register is loaded into the reception 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 disable and then enable I2S before resuming the communication.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side which the received data belongs to. 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 shown as below ***Figure 22-53. I2S master reception disabling sequence***.

Figure 22-53. I2S master reception disabling sequence



I2S slave transmission sequence

The transmission sequence in slave mode is similar to that in master mode. The differences between them are 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 finishes. 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 disable and enable I2S to resume the communication. In slave mode, I2SCH is sensitive to the I2S_WS signal coming from the external master.

In order to disable 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 differences between them are 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 disable I2S, it is mandatory to clear the I2SEN bit immediately after receiving the last RBNE.

I2S full-duplex mode

A single I2S only supports one-way transmission: transmit or receive mode. I2S full-duplex is supported by using an extra I2S module: I2S_ADD simultaneously with I2S. I2S_ADD module has the same function with I2S module, but can only work in slave mode. There are two I2S_ADD modules: I2S_ADD1 and I2S_ADD2, so only I2S1 and I2S2 support full-duplex mode. I2S_ADD's I2S_CK and I2S_WS are internally connected to its respective I2S's respective ports. I2S_ADD's I2S_SD pin is mapped to respective I2S's SPI_MISO pin.

In order to work in full-duplex mode, application should enable the I2S module as well as its corresponding I2S_ADD module. I2S supports two full-duplex modes: master mode and salve mode.

In master full-duplex mode, software should set I2S as a master, and I2S_ADD as a slave. Then I2S_ADD's WS and SCK signals come from the master I2S.

In slave full-duplex mode, software should set both I2S and I2S_ADD as slaves. Then, the WS and CK signals of both I2S_ADD and I2S come from external.

Application may configure I2S into either a transmitter or a receiver and thus, configure I2S_ADD into opposite data direction. During transmission, software should operate registers and handle interrupts for both I2S and I2S_ADD to make a full-duplex transmission.

22.4.6. DMA function

DMA function is the same as SPI mode. The only difference is that the CRC function is not available in I2S mode.

22.4.7. I2S interrupts

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.

- **Transmission buffer empty flag (TBE)**

This bit is set when the transmission buffer is empty, the software can write the next data to the transmission buffer by writing the SPI_DATA register.

- **Reception buffer not empty flag (RBNE)**

This bit is set when reception buffer is not empty, which means that one data is received and stored in the reception buffer, and software can read the data by reading the SPI_DATA register.

- **I2S transmitting ongoing flag (TRANS)**

TRANS is a status flag to indicate whether the transfer is ongoing or not. It is set and cleared by hardware and not controlled by software. This flag will not 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. In the transmission mode, the I2SCH flag is updated every time TBE changes from 0 to 1. In the reception mode, the I2SCH flag is updated every time RBNE changes from 0 to 1. This flag will not generate any interrupt.

Error conditions

There are three error flags:

- **Transmission underrun error flag (TXURERR)**

This situation occurs when the transmission buffer is empty and the valid SCK signal starts in slave transmission mode.

- **Reception overrun error flag (RXORERR)**

This situation occurs when the reception buffer is full and a newly incoming data has been completely received. When overrun occurs, the data in reception buffer is not updated and the newly incoming data is lost.

- **Format error (FERR)**

In slave I2S mode, the I2S monitors the I2S_WS signal and an error flag will be set if I2S_WS toggles at an unexpected position.

I2S interrupt events and corresponding enabled bits are summed up in the ***Table 22-10. I2S interrupt.***

Table 22-10. I2S interrupt

Interrupt flag	Description	Clear method	Interrupt enable bit
TBE	Transmission buffer empty	Write SPI_DATA register	TBEIE
RBNE	Reception buffer not empty	Read SPI_DATA register	RBNEIE

Interrupt flag	Description	Clear method	Interrupt enable bit
TXURERR	Transmission underrun error	Read SPI_STAT register	ERRIE
RXORERR	Reception overrun error	Read SPI_DATA register and then read SPI_STAT register.	
FERR	I2S format error	Read SPI_STAT register	

22.5. Register definition

SPI0 base address: 0x4001 3000

SPI1/I2S1 base address: 0x4000 3800

SPI2/I2S2 base address: 0x4000 3C00

I2S1_ADD base address: 0x4000 3400

I2S2_ADD base address: 0x4000 4000

22.5.1. Control register 0 (SPI_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or 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 data transfers between the MOSI pin of master and the MISO pin of 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: Disable CRC calculation. 1: Enable CRC calculation.
12	CRCNT	CRC next transfer 0: Next transfer is data 1: Next transfer is CRC value

		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 mode 1: Receive-only mode
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. This bit has no meaning in SPI TI mode.
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. This bit has no meaning in SPI TI mode.
7	LF	LSB first mode 0: Transmit MSB first 1: Transmit LSB first This bit has no meaning in SPI TI mode.
6	SPIEN	SPI enable 0: Disable SPI peripheral. 1: Enable SPI peripheral.
5:3	PSC[2:0]	Master clock prescaler selection 000: PCLK/2 001: PCLK/4 010: PCLK/8 011: PCLK/16 100: PCLK/32 101: PCLK/64 110: PCLK/128 111: PCLK/256 PCLK means PCLK2 when using SPI0. PCLK means 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.

22.5.2. Control register 1 (SPI_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or 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								TBEIE	RBNEIE	ERRIE	TMOD	NSSP	NSSDRV	DMATEN	DMAREN
								rw	rw	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: Disable TBE interrupt 1: Enable TBE interrupt. An interrupt is generated when the TBE bit is set.
6	RBNEIE	Receive buffer not empty interrupt enable 0: Disable RBNE interrupt 1: Enable RBNE interrupt. An interrupt is generated when the RBNE bit is set.
5	ERRIE	Errors interrupt enable. 0: Disable error interrupt 1: Enable error interrupt. An interrupt is generated when the CRCERR bit or the CONFERR bit or the FERR bit or the RXORERR bit or the TXURERR bit is set.
4	TMOD	SPI TI mode enable. 0: Disable SPI TI mode 1: Enable SPI TI mode
3	NSSP	SPI NSS pulse mode enable. 0: Disable SPI NSS pulse mode

		1: Enable SPI NSS pulse mode
2	NSSDRV	Drive NSS output 0: Disable master NSS output 1: Enable master NSS output
1	DMATEN	Transmit buffer DMA enable 0: Disable transmit buffer DMA 1: Enable transmit buffer DMA, when the TBE bit in SPI_STAT is set, it will be a DMA request on corresponding DMA channel.
0	DMAREN	Receive buffer DMA enable 0: Disable receive buffer DMA 1: Enable receive buffer DMA, when the RBNE bit in SPI_STAT is set, it will be a DMA request on corresponding DMA channel.

22.5.3. Status register (SPI_STAT)

Address offset: 0x08

Reset value: 0x0000 0002

This register can be accessed by byte (8-bit) or 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							FERR	TRANS	RXORERR	CONFERR	CRCERR	TXURERR	I2SCH	TBE	RBNE
							rc_w0	r	r	r	rc_w0	r	r	r	r

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
8	FERR	Format error SPI TI Mode: 0: No TI mode format error 1: TI mode format error occurs. I2S Mode: 0: No I2S format error 1: I2S format error occurs. This bit is set by hardware and is able to be cleared by writing 0.
7	TRANS	Transmitting ongoing 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 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.

22.5.4. Data register (SPI_DATA)

Address offset: 0x0C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or 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
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 transmission buffer and reception buffer. Write data to SPI_DATA will save the data to transmission buffer and read data from SPI_DATA will get the data from reception 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, transmission buffer and reception 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, transmission buffer and reception buffer are 16-bit.

22.5.5. CRC polynomial register (SPI_CRCPOLY)

Address offset: 0x10

Reset value: 0x0000 0007

This register can be accessed by byte (8-bit) or 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
CRCPOLY[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CRCPOLY[15:0]	CRC polynomial register This register contains the CRC polynomial and it is used for CRC calculation. The default value is 0007h.

22.5.6. RX CRC register (SPI_RCRC)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or 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
RCRC[15:0]															
r															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	RCRC[15:0]	RX CRC value 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 RCRC[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 RCRC[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 in SPI_CTL0 register or the SPIxRST bit in RCU reset register is set.

22.5.7. TX CRC register (SPI_TCRC)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or 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
TCRC[15:0]															
r															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	TCRC[15:0]	TX CRC value 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 TCRC[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 TCRC[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 formats (LF bit of the SPI_CTL0) will get different CRC values.

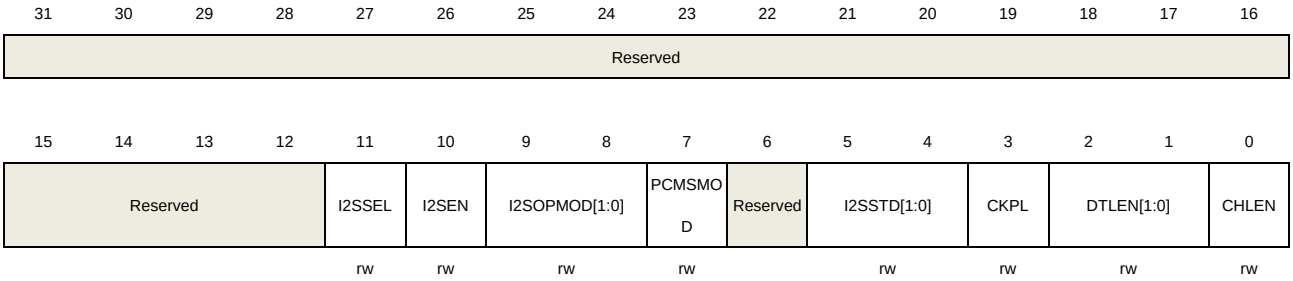
This register is reset when the CRCEN bit in SPI_CTL0 register or the SPIxRST bit in RCU reset register is set.

22.5.8. I2S control register (SPI_I2SCTL)

Address offset: 0x1C

Reset value: 0x0000 0000

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



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: Disable I2S 1: Enable I2S 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.

22.5.9. I2S clock prescaler register (SPI_I2SPSC)

Address offset: 0x20

Reset value: 0x0000 0002

This register can be accessed by byte (8-bit) or 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						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: Disable I2S_MCK output 1: Enable I2S_MCK output 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.

22.5.10. Quad-SPI mode control register (SPI_QCTL) of SPI0

Address offset: 0x80
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or 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													IO23_DRV	QRD	QMOD
													rw	rw	rw

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2	IO23_DRV	Drive IO2 and IO3 enable 0: IO2 and IO3 are not driven in single wire mode. 1: IO2 and IO3 are driven to high in single wire mode. This bit is only available in SPI0.
1	QRD	Quad-SPI mode read select. 0: SPI is in quad wire write mode. 1: SPI is in quad wire read mode. This bit can only be configured when the SPI is not busy (the TRANS bit is cleared).

		This bit is only available in SPI0.
0	QMOD	Quad-SPI mode enable. 0: SPI is in single wire mode. 1: SPI is in Quad-SPI mode. This bit can only be configured when the SPI is not busy (the TRANS bit is cleared). This bit is only available in SPI0.

23. Serial/Quad Parallel Interface (SQPI)

23.1. Overview

Serial/Quad Parallel Interface (SQPI) is a controller for external serial/dual/quad parallel interface memory peripheral. For example: SQPI-PSRAM and SQPI-FLASH.

With this controller, users can use external SQPI interface memory as SRAM simply.

The GD32EPRTxx series chips are internally stacked with PSRAM, the pin connecting to PSARM cannot be used.

23.2. Characteristics

- SQPI controller has two independent sets of configure registers for write operation and read operation.
- SQPI controller support ID length setting.
- SQPI controller can configure the sampling edge of the SQPI_CLK during the read operation.
- SQPI controller support configuring the length of command phase, address phase, and waitcycle phase.
- SQPI controller support configuring output clock frequency which is divided by HCLK.
- SQPI controller support no address phase and data phase operation which is named special command by the controller.
- SQPI controller support READ ID command which is more than 32 bit data during one AHB command.
- SQPI controller support AHB burst operation and 8/16/32 bit AHB command.
- SQPI controller support 256MB external memory space.
  Logic memory address range: 0xB000_0000 - 0xBFFF_FFFF.
- SQPI controller support 6 types mode for different combination of command, address, waitcycle, and data phase. All support modes and IO state are listed below:

  Note: In mode name, the first character indicates command phase valid IO number, the second indicates address phase valid IO number, and the third indicates data phase valid IO number. For each character, S means single (1 IO), D means dual (2 IO), Q means quad (4 IO)

Table 23-1. SQPI Controller Mode Definition

Signal	Direction	Operation Mode					
		SSS	SSQ	SQQ	QQQ	SSD	SDD
SQPI_CLK	Output	Serial Clock					

SQPI_CSN	Output	Chip-Enable (active low)					
Command Phase							
SQPI_D0	Output	O	O	O	O	O	O
SQPI_D1	Output	X	X	X	O	X	X
SQPI_D2	Output	0	0	0	O	0	0
SQPI_D3	Output	1	1	1	O	1	1
Address Phase							
SQPI_D0	Output	O	O	O	O	O	O
SQPI_D1	Output	X	X	O	O	X	O
SQPI_D2	Output	0	0	O	O	0	0
SQPI_D3	Output	1	1	O	O	1	1
Waitcycle Phase							
SQPI_D0	Inout	X	X	X	X	X	X
SQPI_D1	Inout	X	X	X	X	X	X
SQPI_D2	Inout	0	X	X	X	0	0
SQPI_D3	Inout	1	X	X	X	1	1
Data Phase							
SQPI_D0	Inout	O	IO	IO	IO	IO	IO
SQPI_D1	Inout	I	IO	IO	IO	IO	IO
SQPI_D2	Inout	X	IO	IO	IO	X	X
SQPI_D3	Inout	X	IO	IO	IO	X	X

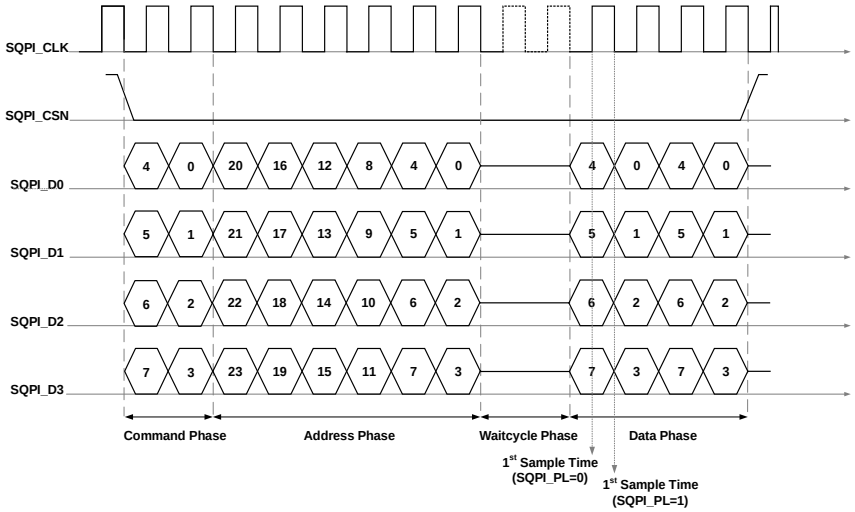
Note: O – Output, I – Input, IO – Inout, 0 – Output 0, 1 – Output 1, X - Hiz

23.3. Function overview

23.3.1. SQPI controller sampling polarity

SQPI controller read operation sampling polarity (SQPI_PL) selection function support user to change the controller sampling time. This function is highly useful when SQPI clock is high. Example showed as below:

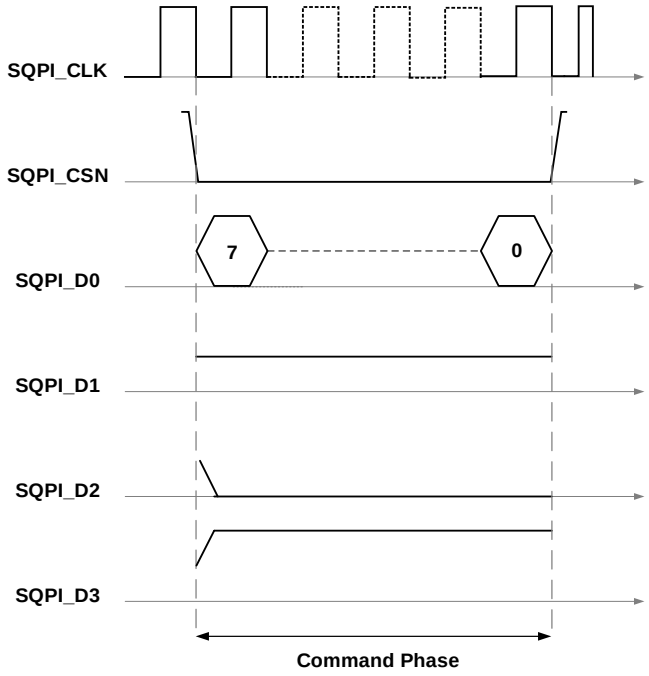
Figure 23-1. SQPI_PL Example



23.3.2. SQPI controller special command

SQPI controller special command (SQPI_SC) function can send only command phase with no address, waitcycle, and data phase. Special command function will be mandatory to SSS mode by hardware. If you set SQPI_SC bit to 1, you must read this bit and wait it cleared before doing other memory access because this can ensure the operation has performed in the interface.

Figure 23-2. SQPI_SC Example

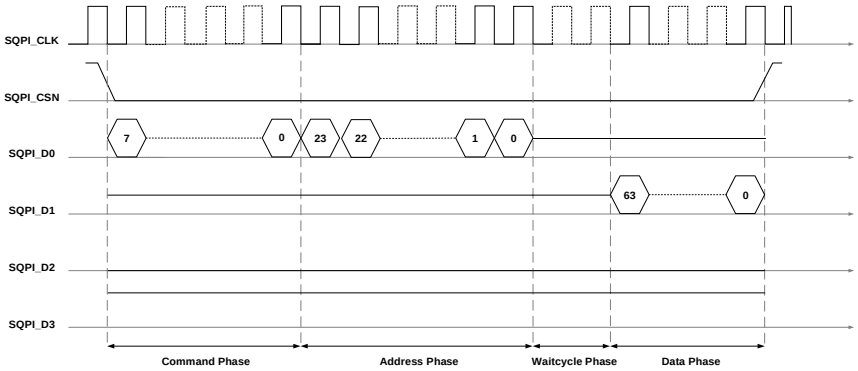


23.3.3. SQPI controller read ID command

For more than 32-bit ID data, SQPI_RDID function can supply help. To use this function, first

you should set SQPI_IDLEN bit to 0x00(64bit,this is default), then set the SQPI_RDID bit to 1 and wait it cleared by hardware through polling this bit, and at last read the SQPI_IDL and SQPI_IDH registers. This command is performed in SSS mode by hardware.

Figure 23-3. SQPI_RDID Example (SQPI_IDLEN=00)



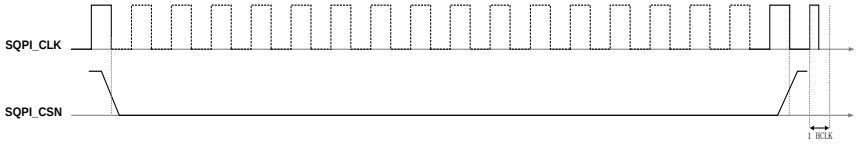
23.3.4. SQPI controller output clock configuration

SQPI clock period is configured by SQPI_CLKDIV bits. The frequency formula of SQPI_CLK is:

$$f_{sqpi\_clk} = \frac{f_{hclk}}{SQPI\_CLKDIV + 1}$$

Note: SQPI_CLKDIV cannot be 0. When SPI_CLKDIV field is even number, the output clock high level time has 1 HCLK period more than low level time. After the rise edge of SQPI_CSN there is 1 HCLK period clock to on SQPI_CLK signal to support some old PSRAM memory.

Figure 23-4. SQPI_CLK Example



23.3.5. SQPI controller initialization

In the beginning, users should program the initialization register SQPI_INIT. Data sampling clock edge is selected via the SQPI_PL bit, read device ID length could be configured by the SQPI_IDLEN bits, address bit number is controlled by the SQPI_ADDRBIT, command bit number is set by SQPI_CMDBIT, and the SQPI controller clock is configured by SQPI_CLKDIV bits.

23.3.6. Read ID command flow

The first, user should configure SQPI_RCMD bits by Read ID command (e.g. 0x9F for SQPIPSRAM) and read waitcycle number in SQPI_RCMD register. The second user sets SQPI_RID bit to 1 and wait it reset to 0. The third, user can get ID value by read SQPI_IDL and SQPI_IDH registers.

23.3.7. Read/Write operation flow

Six modes of memory access are possible. Access mode should be configured before read/write operations. Read/Write command mode is programmed by the SQPI_RMODE and SQPI_WMODE, wait cycle is controlled by the SQPI_RWAITCYCLE and SQPI_WWAITCYCLE bit, and the specific memory operating command should be programmed in SQPI_RCMD and SQPI_WCMD bit, these read/write settings are located in SQPI_RCMD and SQPI_WCMD registers respectively.

After memory access mode configuration, user can directly access external device as SRAM by using SQPI memory logic address.

23.3.8. SQPI controller mode timing

SQPI controller mode timing for read/write operation, each AHB read/write access to SQPI memory logic address will transfer to one of below timing:

Figure 23-5. SQPI SSS Mode Timing (SPI)

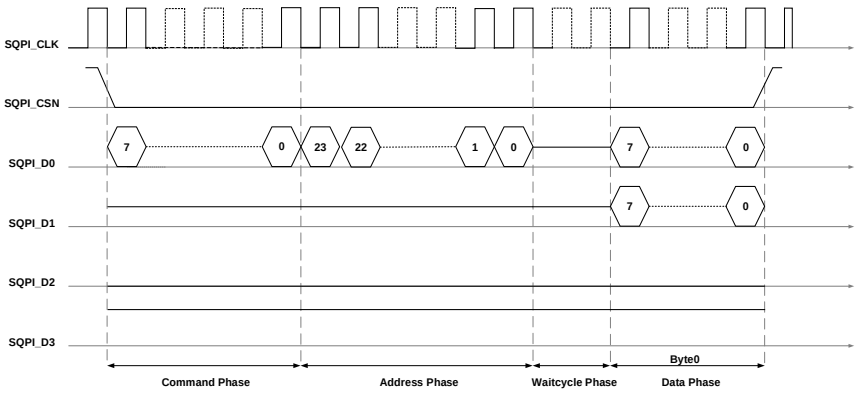


Figure 23-6. SQPI SSQ Mode Timing

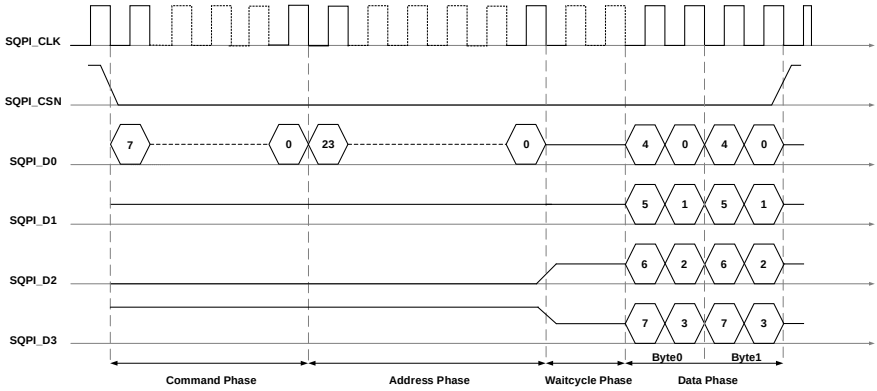


Figure 23-7. SQPI SQQ Mode Timing (SQPI)

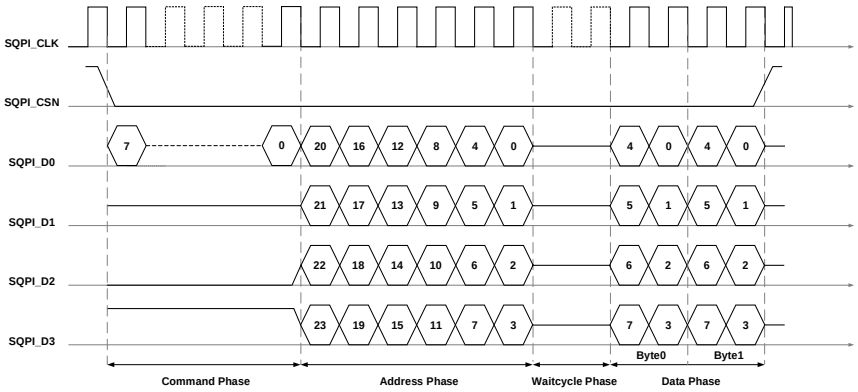


Figure 23-8. SQPI QQQ Mode Timing (QPI)

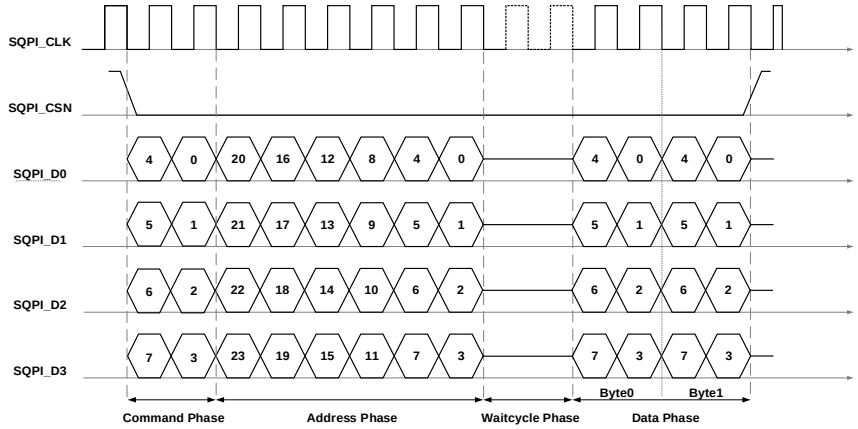


Figure 23-9. SQPI SSD Mode Timing

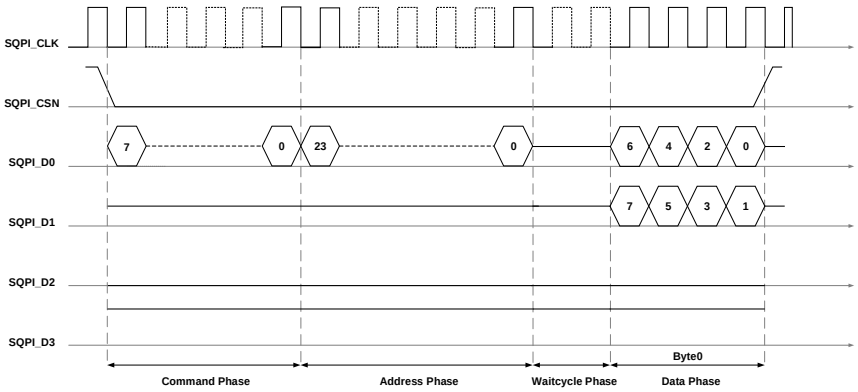
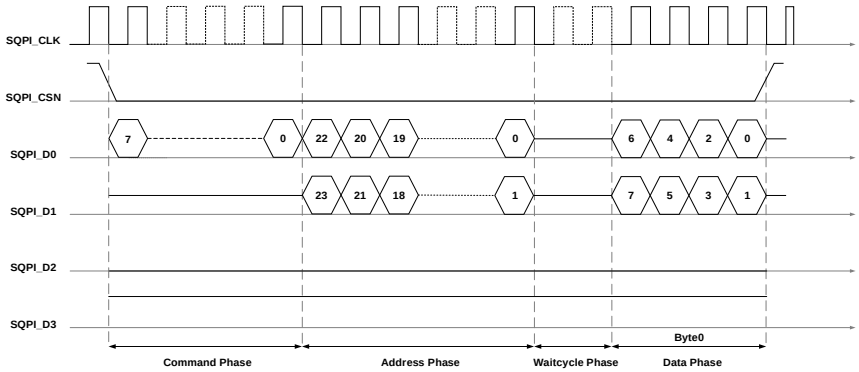


Figure 23-10. SQPI SDD Mode Timing



23.4. Register definition

SQPI start address: 0xA000 1000

23.4.1. SQPI Initial Register (SQPI_INIT)

Address offset: 0x00

Reset Value: 0x1805 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
SQPI_PL	SQPI_IDLEN[1:0]		SQPI_ADDRBIT[4:0]					SQPI_CLKDIV[5:0]						SQPI_CMDBIT[1:0]	
rw	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	SQPI_PL	Read data sample polarity. 0: Sample data at rising edge(default) 1: Sample data at falling edge.
30:29	SQPI_IDLEN[1:0]	SQPI controller external memory ID length. 00:64-bit 01:32-bit 10:16-bit 11:8-bit
28:24	SQPI_ADDRBIT[4:0]	Bit number of SPI PSRAM address phase. Default：24
23:18	SQPI_CLKDIV[5:0]	Clock divider for SQPI output clock.

		0x0 is invalid. Output clock frequency is f_{hclk}/(SQPI_CLKDIV+1) Note: When SQPI_CLKDIV field is even number, the output clock high level time has 1 HCLK period more than low level time.
17:16	SQPI_CMDBIT[1:0]	Bit number of SQPI controller command phase 00: 4 bit 01: 8 bit (default) 10: 16 bit 11: Reserved
15:0	Reserved	Must be kept at reset value.

23.4.2. SQPI Read Command Register (SQPI_RCMD)

Address offset: 0x04

Reset value: 0x0010 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
SQPI_RID	Reserved								SQPI_RMODE[2:0]			SQPI_RWAITCYCLE[3:0]			
rw									rw			rw			

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

Bits	Fields	Descriptions
31	SQPI_RID	Send read ID command, command code comes from SQPI_RCMD.
30:23	Reserved	Must be kept at reset value.
22:20	SQPI_RMODE[2:0]	SQPI controller read command mode: 000: SSQ mode 001: SSS mode 010: SQQ mode 011: QQQ mode 100: SSD mode 101: SDD mode
19:16	SQPI_RWAITCYCLE[3:0]	SQPI read command waitcycle number after address phase

15:0	SQPI_RCMD[15:0]	SQPI read command for AHB read transfer SQPI_RCMD[3:0] are valid when SQPI_CMDBIT=00 SQPI_RCMD[7:0] are valid when SQPI_CMDBIT=01 SQPI_RCMD[15:0] are valid when SQPI_CMDBIT=10

Note: Before write 1 to SQPI_RID bit, you must ensure it is cleared and after set SQPI_RID to 1, you must wait SQPI_RID cleared.

23.4.3. SQPI Write Command Register (SQPI_WCMD)

Address offset: 0x08
Reset value: 0x0010 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
SQPI_SC	Reserved								SQPI_WMODE [2:0]			SQPI_WWAITCYCLE [3:0]			
rs									rw			rw			

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

Bits	Fields	Descriptions
31	SQPI_SC	Send special command which does not have address and data phase, command code comes from SQPI_WCMD.
30:23	Reserved	Must be kept at reset value.
22:20	SQPI_WMODE[2:0]	SQPI controller write command mode: 000: SSQ mode 001: SSS mode 010: SQQ mode 011: QQQ mode 100: SSD mode 101: SDD mode
19:16	SQPI_WWAITCYCLE [3:0]	SQPI write command waitcycle number after address phase
15:0	SPI_WCMD[15:0]	SQPI write command for AHB write transfer

Note: Before write 1 to SQPI_SC bit, you must ensure it is cleared and after set SQPI_SC to 1, you must wait SQPI_SC cleared

23.4.4. SQPI ID Low Register (SQPI_IDL)

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
SQPI_IDL [31:16]															
r															

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

Bits	Fields	Descriptions
31:0	SQPI_IDL[31:0]	ID Low Data saved for SQPI Read ID Command SQPI_IDL[15:0] is valid when SQPI_IDLEN=10 SQPI_IDL[7:0] is valid when SQPI_IDLEN=11.

23.4.5. SQPI ID High Register (SQPI_IDH)

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
SQPI_IDH[31:16]															
r															

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

Bits	Fields	Descriptions
31:0	SQPI_IDH[31:0]	ID High Data saved for SQPI read ID command This register only valid when SQPI_IDLEN = 00.

24. Secure digital input/output interface (SDIO)

24.1. Introduction

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***.

24.2. Main features

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.

24.3. 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.

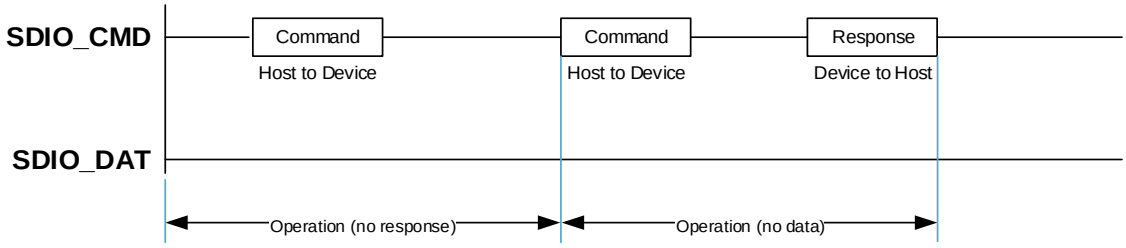
There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case no data token is present in an operation. 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 24-1. 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 24-1. 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 24-2. SDIO multiple blocks read operation*** is the multiple blocks read operation and ***Figure 24-3. SDIO multiple blocks write operation*** is the multiple blocks 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 24-2. SDIO multiple blocks read operation

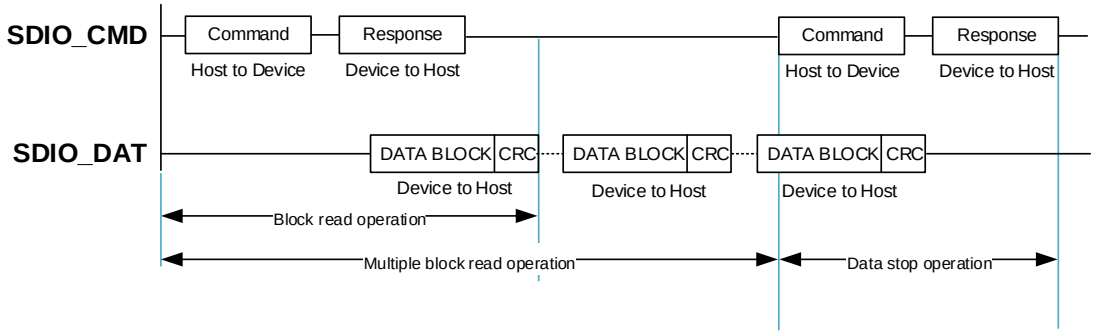
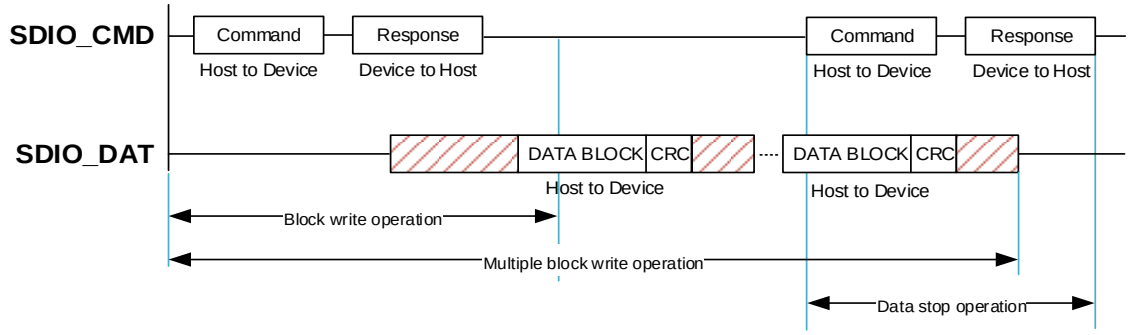


Figure 24-3. SDIO multiple blocks write operation



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 24-4. SDIO sequential read operation*** and ***Figure 24-5. SDIO sequential write operation*** are the stream read and write operation.

Figure 24-4. SDIO sequential read operation

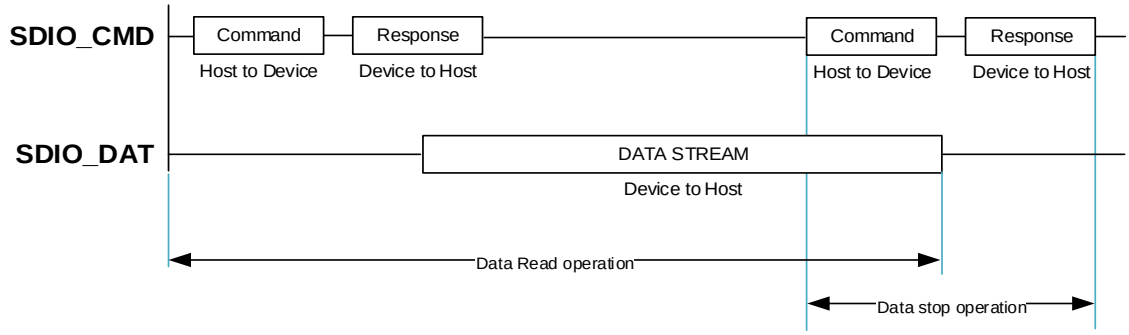
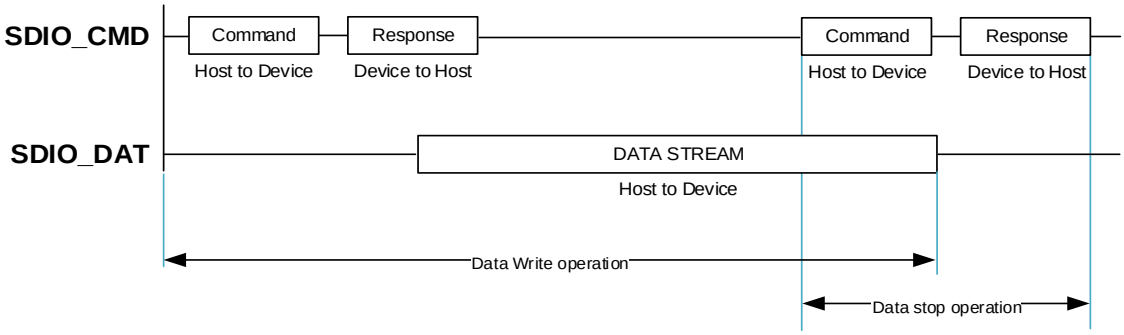


Figure 24-5. SDIO sequential write operation

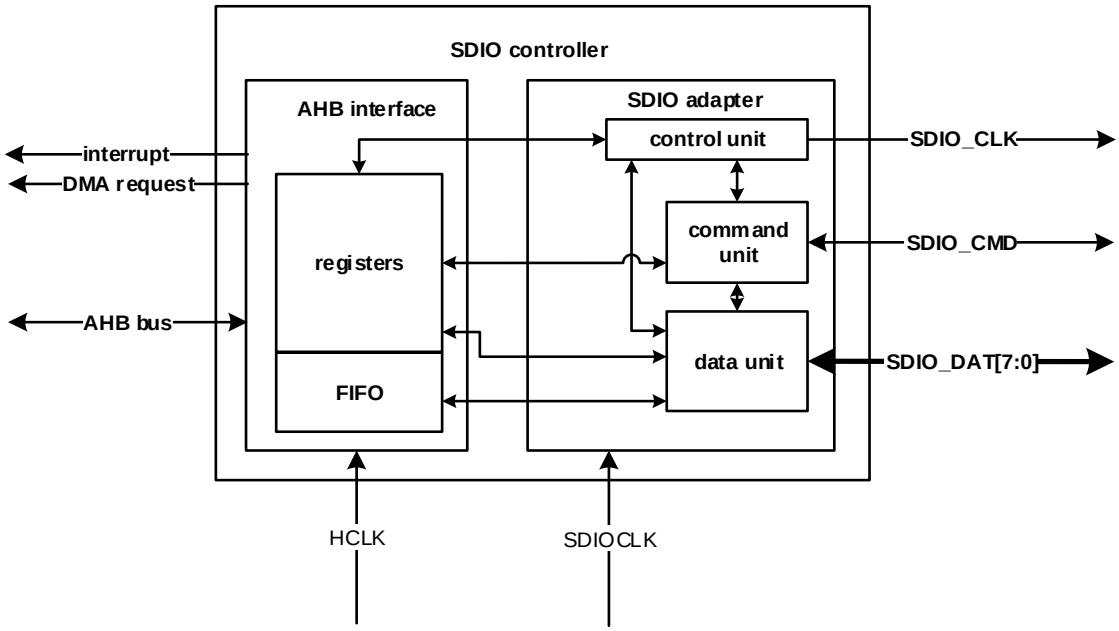


24.4. SDIO functional description

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 24-6. SDIO block diagram



24.4.1. 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)

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 24-1. 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

24.4.2. 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.

24.5. Card functional description

24.5.1. 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.

24.5.2. 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 24-7. Command Token Format***, needing a transmission time of 1.92μs (25 MHz) 0.96μs (50 MHz) and 0.92us (52 MHz).

Figure 24-7. Command Token Format

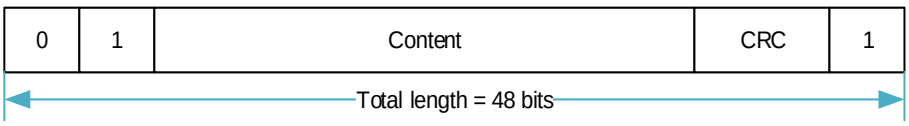


Table 24-2. Command format

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 24-3. Card command classes (CCCs)***). Each class supports a set of card functionalities. ***Table 24-3. 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 24-3. 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 24-4. 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 24-5. 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 24-6. 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 24-7. 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 24-5. 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.

Cmd index	type	argument	Response format	Abbreviation	Description
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 24-8. 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_GROUP_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_GROUP_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 24-9. 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 24-10. 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 24-5. 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 24-11. 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 Count Other bits are reserved bits.	R1(read)/ R1b(write)	RW_MULTIPLE_REGISTER	Read or write register in address range.
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 24-12. 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		IO_RW_EXTENDED	This command allows the reading or writing of a large number of I/O registers with a single command.

Cmd index	type	argument	Response format	Abbreviation	Description
		Count			
Note: 1.CMD39, CMD40 are only for MMC. 2. CMD52, CMD53 are only for SD I/O card.					

Table 24-13. 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).

24.5.3. 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.

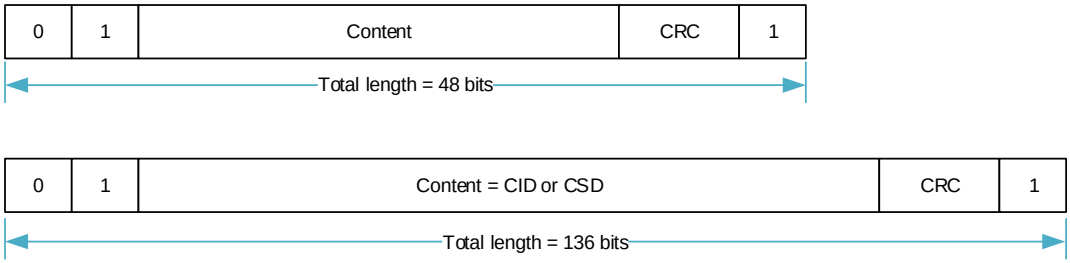
- **R7 :** Card interface condition.

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 24-8. 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 24-8. 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 24-14. 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 index	card status	CRC7	end bit

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 24-15. 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 24-16. 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 24-17. 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 bit	transmission bit	CMD39	RCA [31:16]	status [15]	register address [14:8]	read register contents [7:0]	CRC7	end 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 24-18. 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 24-19. 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 24-20. 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'	'11010X'	'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 24-21. 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 24-22. 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

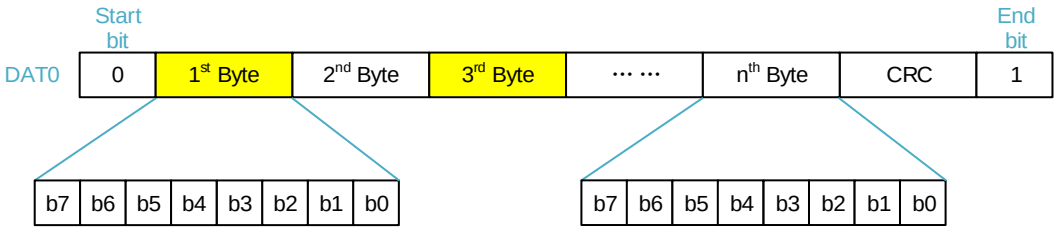
24.5.4. Data packets format

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 24-9. 1-bit data bus width***, ***Figure 24-10. 4-bit data bus width*** and ***Figure 24-11. 8-bit data bus width*** show the data packet format when data bus wide is 1-bit, 4-bit and 8-bit.

Figure 24-9. 1-bit data bus width



4-bit data packet format

Figure 24-10. 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 24-11. 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	b3	b3	b3				··· ···		b3	CRC	1
DAT2	0	b2	b2	b2				··· ···		b2	CRC	1
DAT1	0	b1	b1	b1				··· ···		b1	CRC	1
DAT0	0	b0	b0	b0				··· ···		b0	CRC	1

24.5.5. 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 24-23. 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_FAIL	ERX	’0’= no error	Set when a sequence or	C

	ED		’1’= error	password error has been detected in lock/unlock card command.	
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 internal ECC.	A
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:	CURRENT_STATE	SX	0 = idle	The state of the card when	B

9]			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	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.	
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 24-24. 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
[407:402]	ERASE_TIMEOUT	SR	Timeout value for erasing areas specified by UNIT_OF_ERASE_AU	(See below)	A
[401:400]	ERASE_OFFSET	SR	Fixed offset value added to erase	(See below)	A

Bits	Identifier	Type	Value	Description	Clear Condition
			time.		
[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 24-25. Performance move field***.

Table 24-25. Performance move field

PERFORMANCE_MOVE	Value Definition
00h	Sequential Write
01h	1 [MB/sec]
02h	2 [MB/sec]
.....	.....
FEh	254 [MB/sec]

PERFORMANCE_MOVE	Value Definition
FFh	Infinity

AU_SIZE

This 4-bit field indicates AU Size and the value can be selected from 16 KB.

Table 24-26. 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 24-26. AU_SIZE field***. The card can set any AU size specified in ***Table 24-27. Maximum AU size*** that is less than or equal to the maximum AU size. The card should set smaller AU size as possible.

Table 24-27. 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 N_{ERASE}. When N_{ERASE} 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 24-28. Erase size field

ERASE_SIZE	Value Definition
0000h	Erase Time-out Calculation is not supported.
0001h	1 AU
0002h	2 AU

ERASE_SIZE	Value Definition
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.

$$\text{Erase timeout of X AU} = \frac{T_{ERASE}}{N_{ERASE}} * X + T_{OFFSET} \quad (24\text{-}1)$$

Table 24-29. 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 24-30. Erase offset field

ERASE_OFFSET	Value Definition
0h	0 [sec]
1h	1 [sec]
2h	2 [sec]
3h	3 [sec]

24.6. Programming sequence

24.6.1. 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.

24.6.2. 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.

24.6.3. 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.

24.6.4. 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.

24.6.5. 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:

$$\max\ \text{write}\ \ \text{frequence} = \min\left(\text{TRAN\_SPEED}, \frac{8*2^{\text{WRITE\_BL\_LEN}}-100*\text{NSAC}}{\text{TAAC}*\text{R2W\_FACTOR}}\right) \quad (24\text{-}2)$$

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:

$$\max\ \text{read}\ \ \text{frequence} = \min\left(\text{TRAN\_SPEED}, \frac{8*2^{\text{READ\_BL\_LEN}}-100*\text{NSAC}}{\text{TAAC}*\text{R2W\_FACTOR}}\right) \quad (24\text{-}3)$$

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

24.6.6. 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.

24.6.7. 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).

24.6.8. 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***.

24.6.9. 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 24-31. Lock***

card data structure describes the structure of the command data block.

Table 24-31. 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.

24.7. Specific operations

24.7.1. 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 24-12. Read wait control by stopping SDIO_CLK*** and ***Figure 24-13. Read wait operation using SDIO_DAT[2]*** show the Read Wait mode about stop the SDIO_CLK and use SDIO_DAT[2].

Figure 24-12. Read wait control by stopping SDIO_CLK

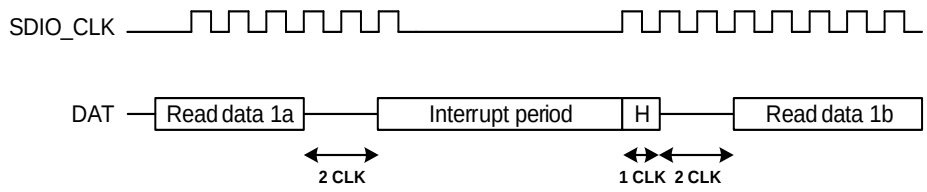
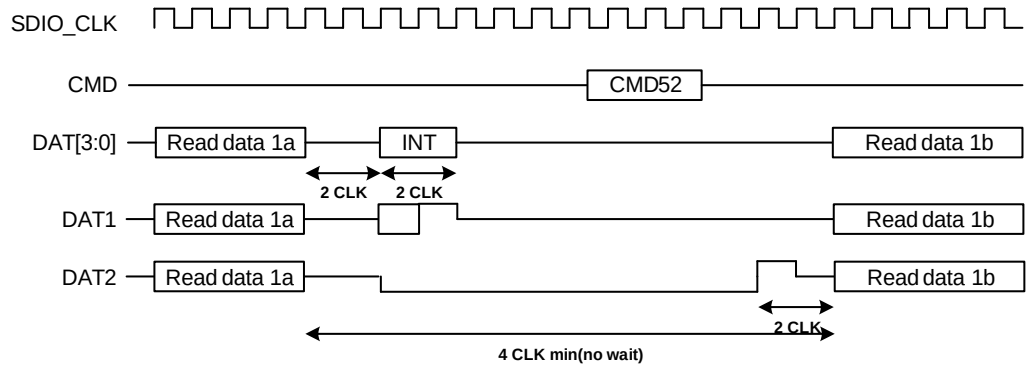


Figure 24-13. Read wait operation using SDIO_DAT[2]



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

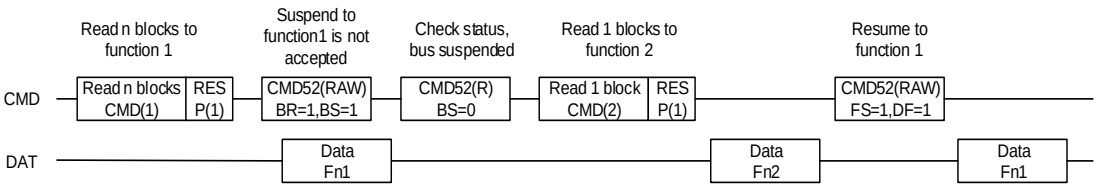
SDIO_DATACTL[8] = 1) and data direction is from card to SD I/O (SDIO_DATACTL[1] = 1), the DSM directly moves from Idle to Read Wait. In Read Wait the DSM drives SDIO_DAT[2] to 0 after 2 SDIO_CLK clock cycles. In this state, when you set the RWSTOP bit (SDIO_DATACTL[9]), the DSM remains in Wait for two 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 24-14. 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 24-14. 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 24-15. Read Interrupt cycle timing shows the timing of the interrupt period for single data transaction read cycles.

Figure 24-15. Read Interrupt cycle timing

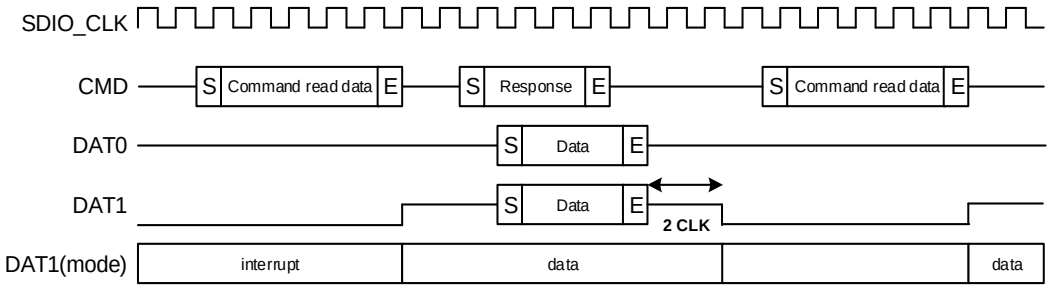
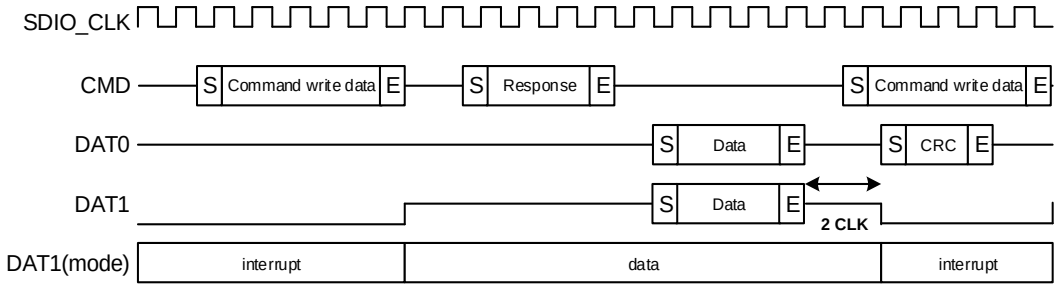


Figure 24-16. Write interrupt cycle timing



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 24-17. Multiple block 4-Bit read interrupt cycle timing* shows the operation for an interrupt during a 4-bit multi-block read and *Figure 24-18. Multiple block 4-Bit write interrupt cycle timing* shows the operation for an interrupt

during a 4-bit multi-block write.

Figure 24-17. Multiple block 4-Bit read interrupt cycle timing

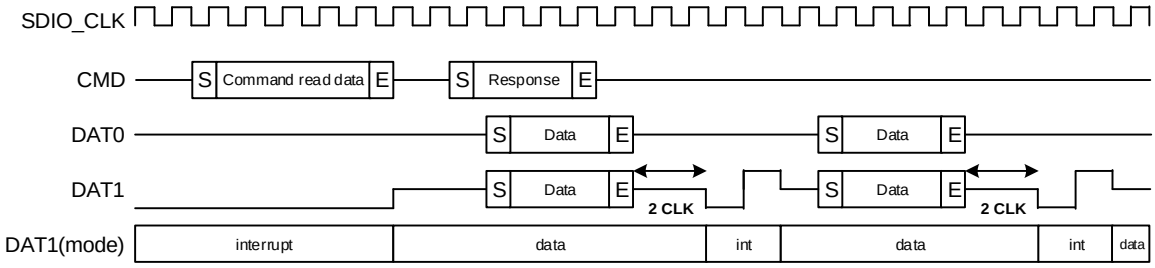
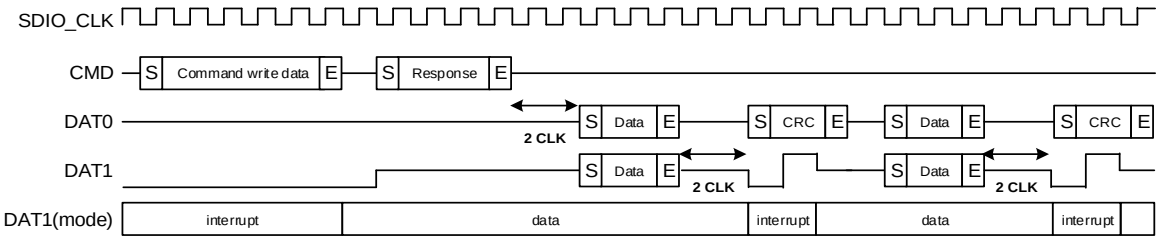


Figure 24-18. Multiple block 4-Bit write interrupt cycle timing



24.7.2. 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.

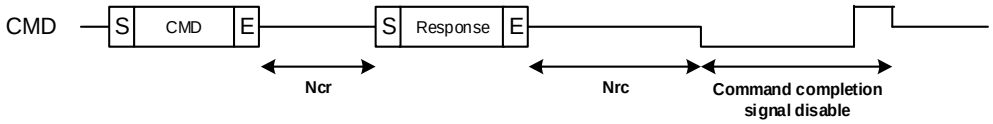
After the host detects a command completion signal from the device, it should issue a FAST_IO (CMD39) command to read the ATA Status register to determine the ending status for the ATA command.

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 24-19. The operation for command completion disable signal



24.8. SDIO registers

SDIO base address: 0x4001 8000

24.8.1. 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.

24.8.2. 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
DIV[8]	Reserved														
rw															

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	DIV[8]	MSB of Clock division This field defines the MSB division between the input clock (SDIOCLK) and the output clock, refer to bit 7:0 of SDIO_CLKCTL.
30:15	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.

24.8.3. 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)

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

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

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.

24.8.4. 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)

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	ATAEN	NINTEN	ENCMDC	SUSPEND	CSMEN	WAITDEND	INTWAIT	CMDRESP[1:0]		CMDIDX[5:0]					
	rw	rw	rw	rw	rw	rw	rw	rw		rw					

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

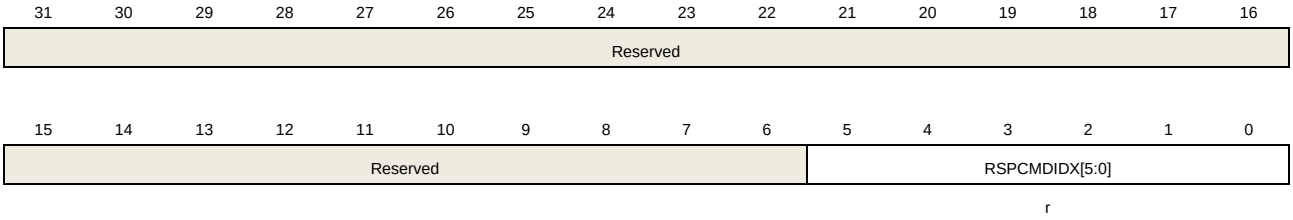
		1: CE-ATA enable
13	NINTEN	No CE-ATA Interrupt (CE-ATA only) This bit defines if there is CE-ATA interrupt or not. This bit is only used when CE-ATA card. 0: CE-ATA interrupt enable 1: CE_ATA interrupt disable
12	ENCMDC	CMD completion signal enabled (CE-ATA only) This bit defines if there is command 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.

24.8.5. Command index response register (SDIO_RSPCMDIDX)

Address offset: 0x10
Reset value: 0x0000 0000

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

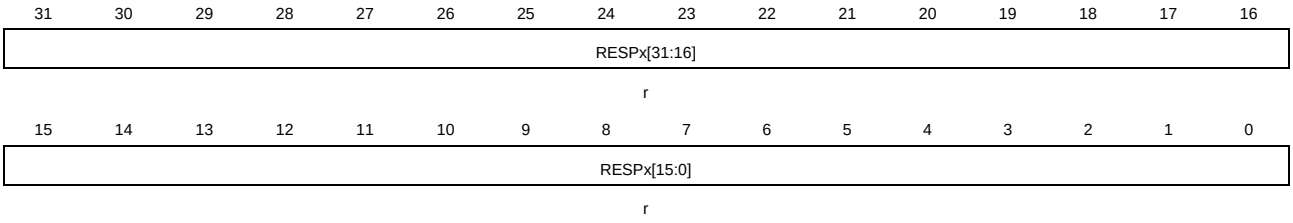


Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5:0	RSPCMDIDX[5:0]	Last response command index Read-only bits field. This field contains the command index of the last command response received. If the response doesn't have the command index (long response and short response of R3), the content of this register is undefined.

24.8.6. 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)



Bits	Fields	Descriptions
31:0	RESPx[31:0]	Card state. The content of the response, see *Table 24-32. 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 24-32. 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]

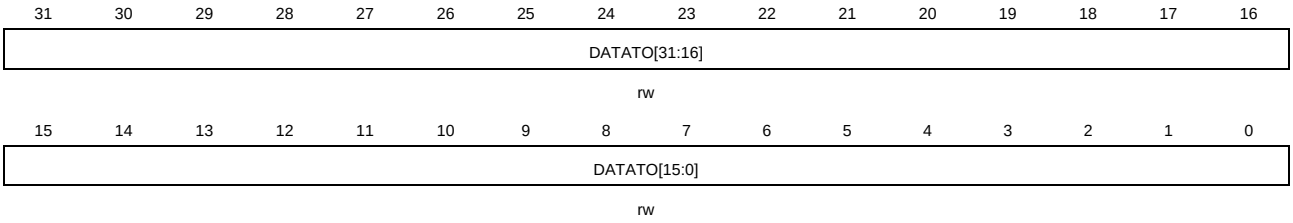
Register	Short response	Long response
SDIO_RESP2	reserved	Card response [63:32]
SDIO_RESP3	reserved	Card response [31:1],plus bit 0

24.8.7. Data timeout register (SDIO_DATATO)

Address offset: 0x24

Reset value: 0x0000 0000

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



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.

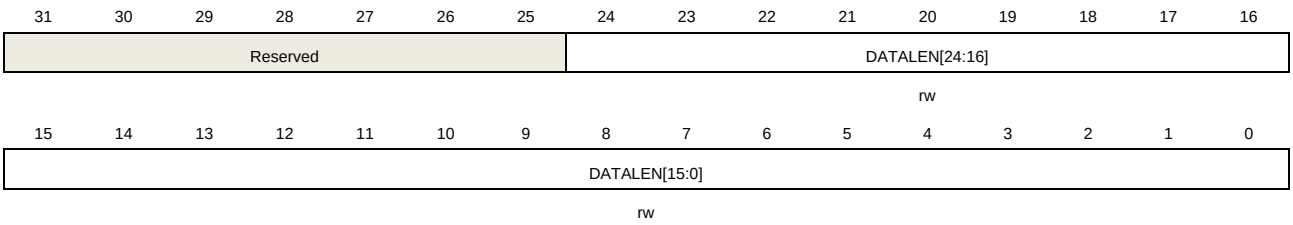
Note: The data timer register and the data length register must be updated before being written to the data control register when need a data transfer.

24.8.8. Data length register (SDIO_DATALEN)

Address offset: 0x28

Reset value: 0x0000 0000

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



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.

24.8.9. 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	TRANSMOD	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

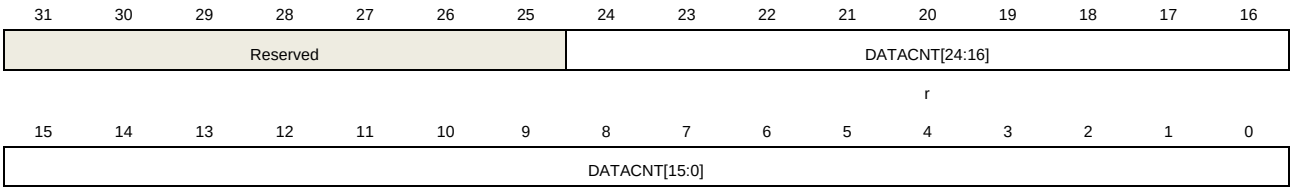
		0101: block size = 2^5 = 32 bytes 0110: block size = 2^6 = 64 bytes 0111: block size = 2^7 = 128 bytes 1000: block size = 2^8 = 256 bytes 1001: block size = 2^9 = 512 bytes 1010: block size = 2^{10} = 1024 bytes 1011: block size = 2^{11} = 2048 bytes 1100: block size = 2^{12} = 4096 bytes 1101: block 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.

24.8.10. 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)



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.

24.8.11. 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)

24.8.12. 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.

24.8.13. 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

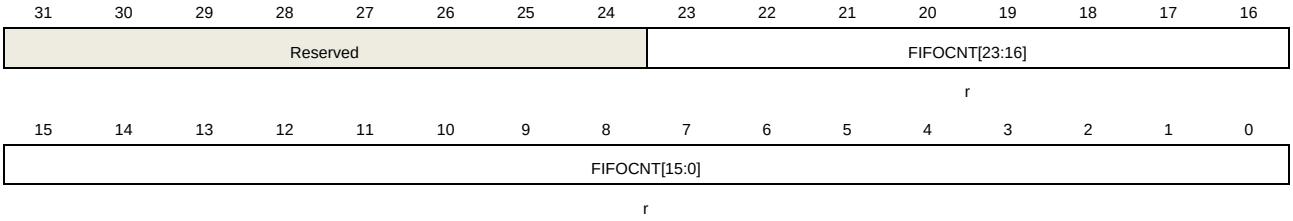
		Write 1 to this bit to enable the interrupt.
9	STBITEIE	Start bit error interrupt enable Write 1 to this bit to enable the interrupt.
8	DTENDIE	Data end interrupt enable Write 1 to this bit to enable the interrupt.
7	CMDSENDIE	Command sent interrupt enable Write 1 to this bit to enable the interrupt.
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.

24.8.14. FIFO counter register (SDIO_FIFOCNT)

Address offset: 0x48

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:0	FIFOCNT[23:0]	FIFO counter.

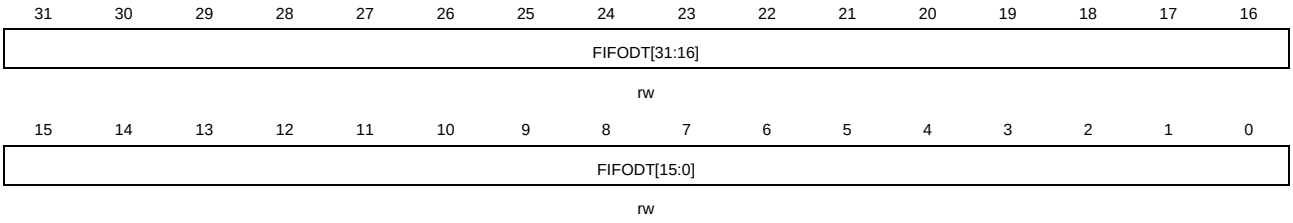
These bits define the remaining number words to be written or read from the FIFO. It loads the data length register (SDIO_DATALEN[24:2] if SDIO_DATALEN is word-aligned or SDIO_DATALEN[24:2]+1 if SDIO_DATALEN is not word-aligned) when DATAEN is set, and start count decrement when a word write to or read from the FIFO.

24.8.15. 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)



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.

25. External memory controller (EXMC)

25.1. Overview

The external memory controller EXMC, is used as a translator for MCU to access a variety of external memory. By configuring the related registers, it can automatically convert AMBA memory access protocol into a specific memory access protocol, such as SRAM, ROM, NOR Flash, NAND Flash and PC Card. Users can also adjust the timing parameters in the configuration registers to improve 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.

25.2. Characteristics

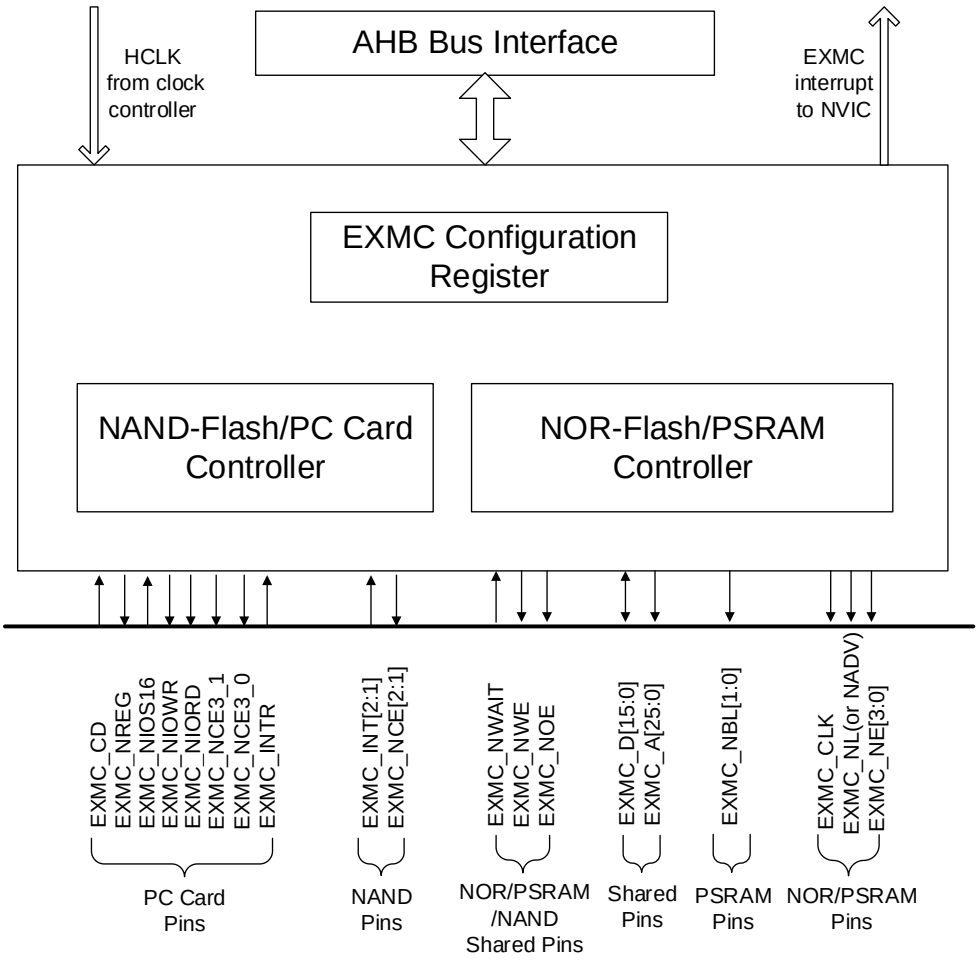
- Supported external memory:
  - SRAM
  - PSRAM
  - ROM
  - NOR Flash
  - 8-bit or 16-bit NAND Flash
  - 16-bit PC Card
- Protocol translation between the AMBA and the multitude of external memory protocol.
- Offering a variety of programmable timing parameters to meet user's specific needs.
- Each bank 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.

25.3. Function overview

25.3.1. 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 25-1. The EXMC block diagram



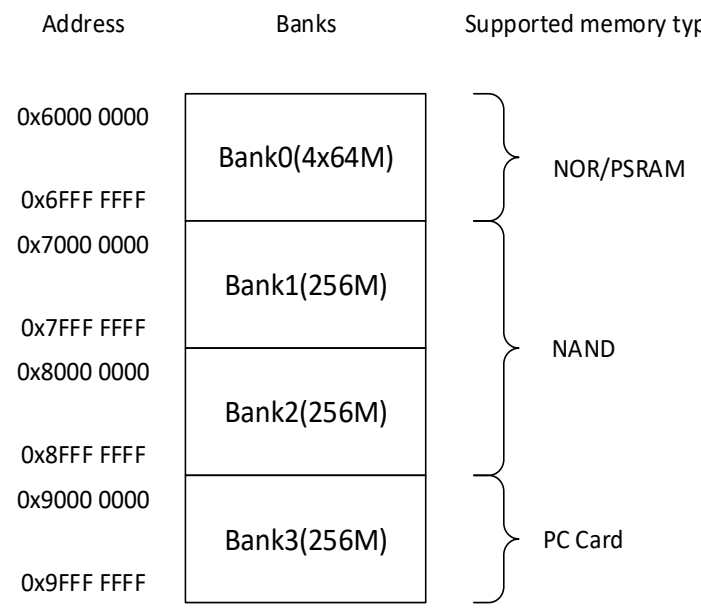
25.3.2. Basic regulation of EXMC access

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 follows 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 will 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.

25.3.3. External device address mapping

Figure 25-2. 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. Bankx(x=1, 2) is 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.

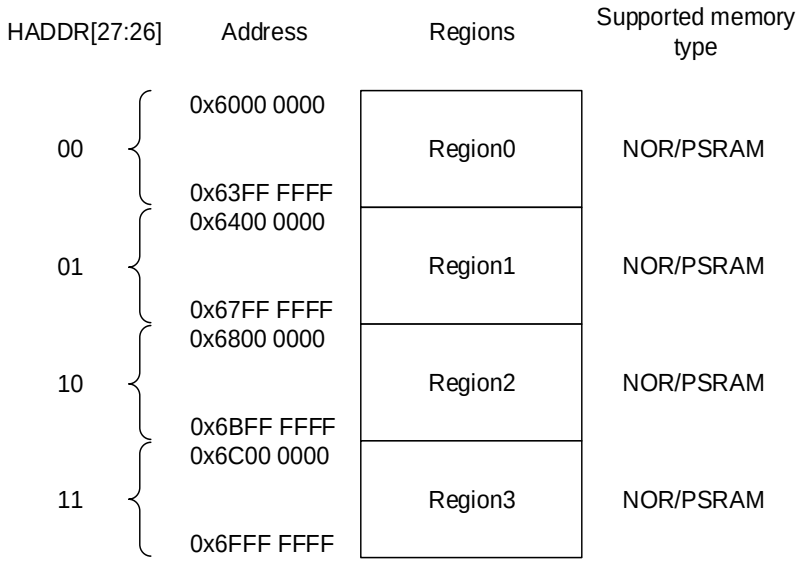
Bank1 and bank2 are used to access NAND Flash exclusively.

Bank3 is used for PC Card access.

NOR/PSRAM address mapping

Figure 25-3. Four regions of bank0 address mapping reflects the address mapping of the four regions of bank0. Internal AHB address lines HADDR[27:26] bit are used to select the four regions.

Figure 25-3. Four regions of bank0 address mapping



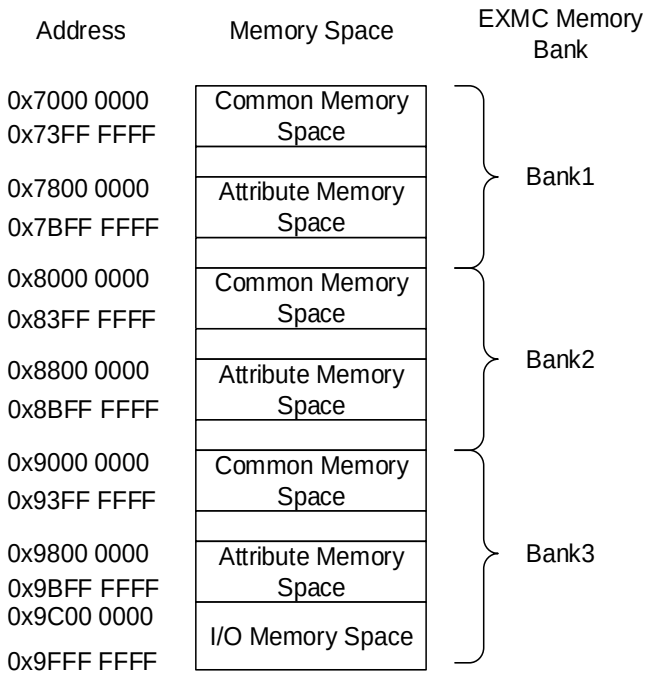
HADDR[25:0] is the byte address whereas the external memory may not be byte accessed, this will lead to address inconsistency. EXMC can adjust HADDR to accommodate the data width of the external memory according to the following rules.

- 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 25-4. NAND/PC Card address mapping***.

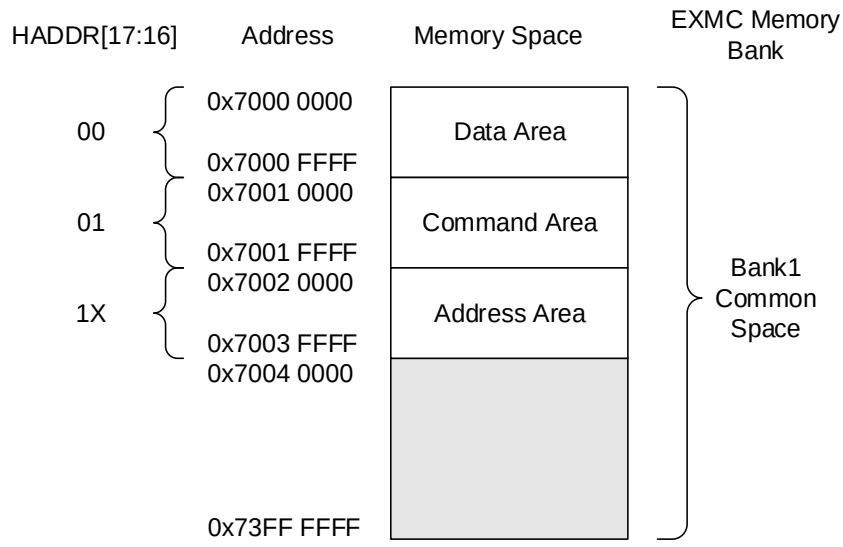
Figure 25-4. NAND/PC Card address mapping



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 25-5. Diagram of bank1 common space.***

Figure 25-5. 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.

25.3.4. 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 25-1. 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_D[15:0]	Input/output	Async/Sync (muxed)	Address/Data bus

EXMC Pin	Direction	Mode	Functional description
	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 25-2. 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 25-3. 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
	Sync	R	16	16	
	Sync	R	32	16	
PSRAM	Async	R	8	16	
	Async	W	8	16	Use of byte lanes

Memory	Access Mode	R/W	AHB Transaction Size	Memory Transaction Size	Comments
					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 25-4. 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

Parameter	Function	Access mode	Unit	Min	Max
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 25-5. 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 25-5. EXMC_timing models***, EXMC NOR Flash / PSRAM controller provides a variety of timing model, users can modify those parameters listed in ***Table 25-4. 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 25-6. Mode 1 read access

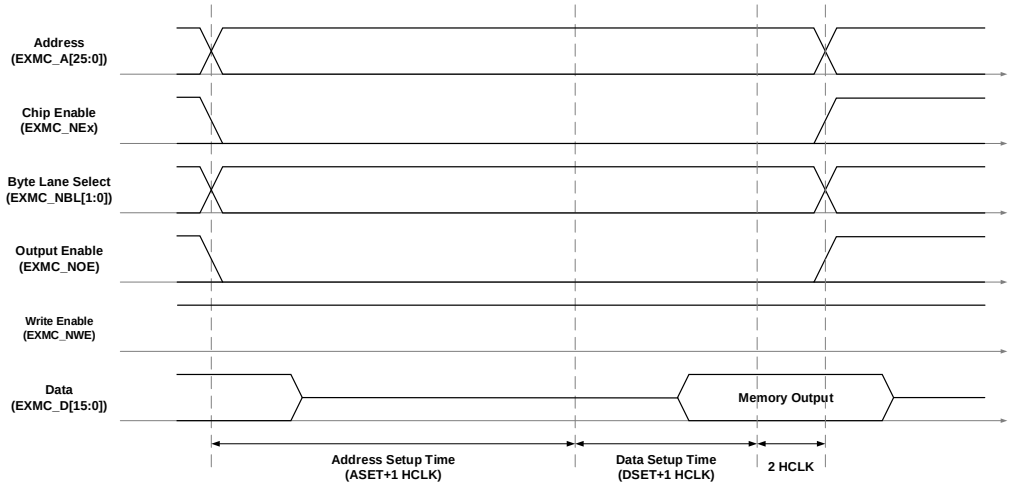


Figure 25-7. Mode 1 write access

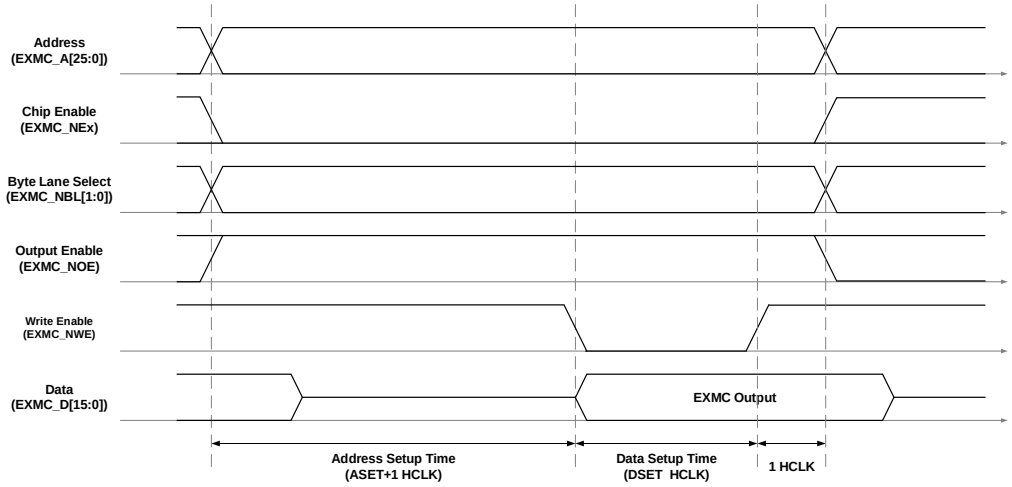


Table 25-6. Mode 1 related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	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

Bit Position	Bit Name	Reference Setting Value
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 25-8. Mode A read access

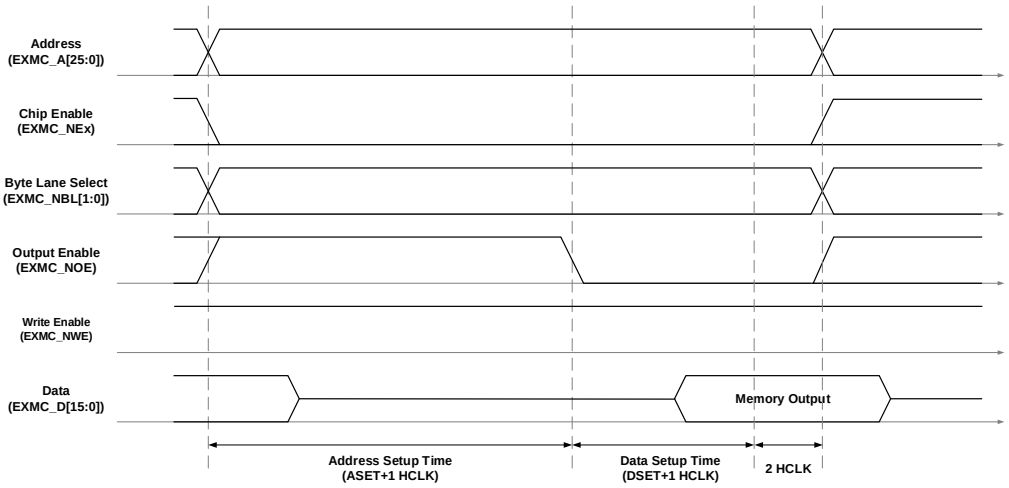
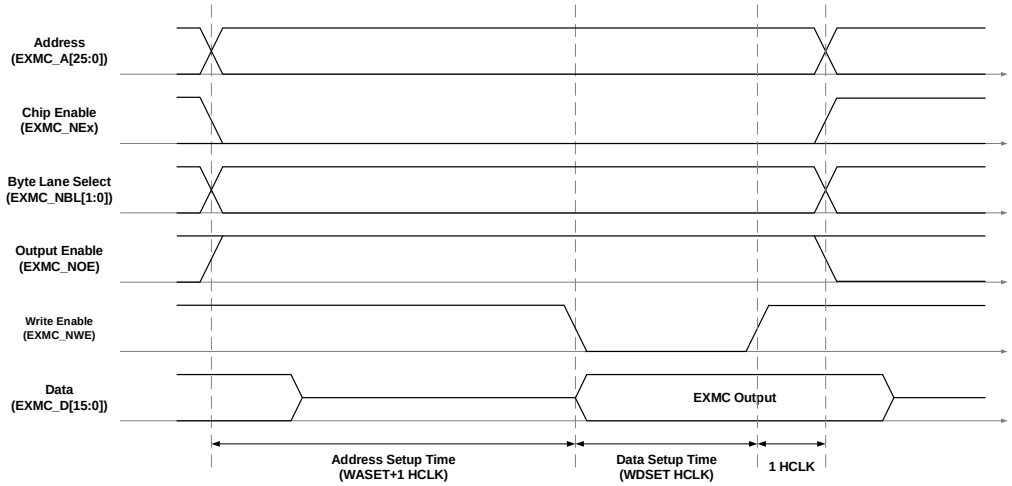


Figure 25-9. Mode A write access



The difference 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 25-7. Mode A related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWAIT	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

Bit Position	Bit Name	Reference Setting Value
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
27-20	Reserved	0x00
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
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 25-10. Mode 2/B read access

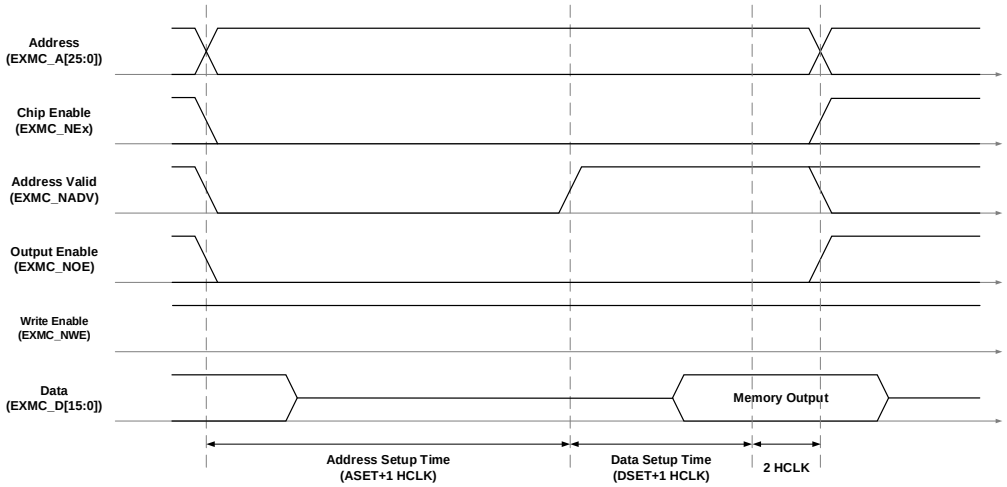


Figure 25-11. Mode 2 write access

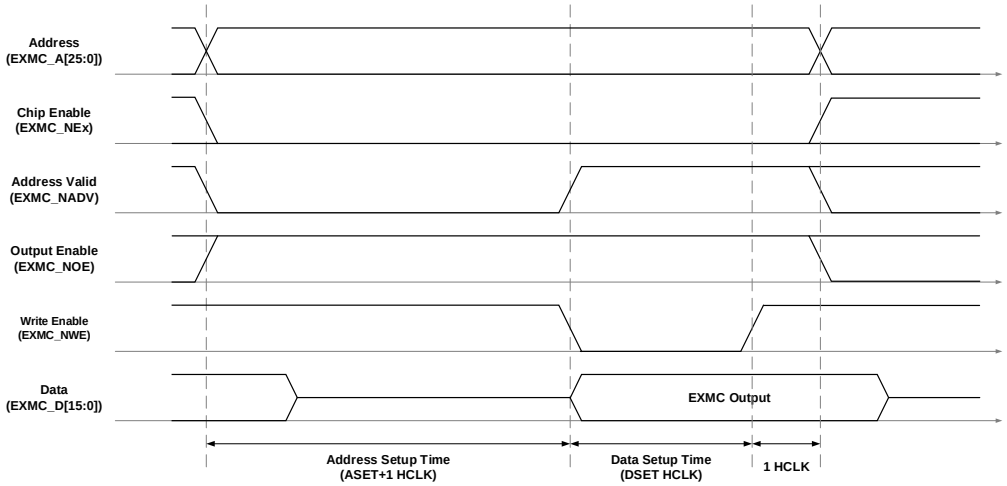


Figure 25-12. Mode B write access

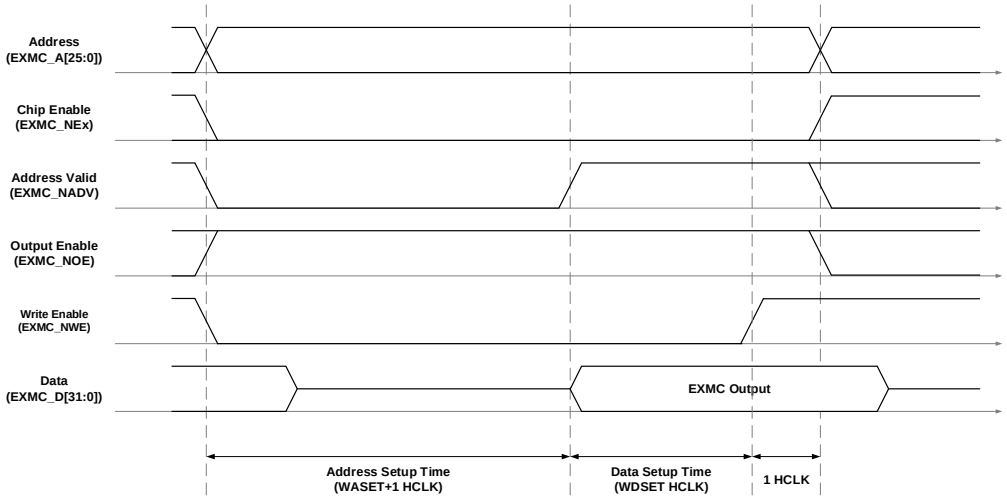


Table 25-8. Mode 2/B related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx(Mode 2, Mode B)		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWAIT	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

Bit Position	Bit Name	Reference Setting Value
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
27-20	Reserved	0x00
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
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 25-13. Mode C read access

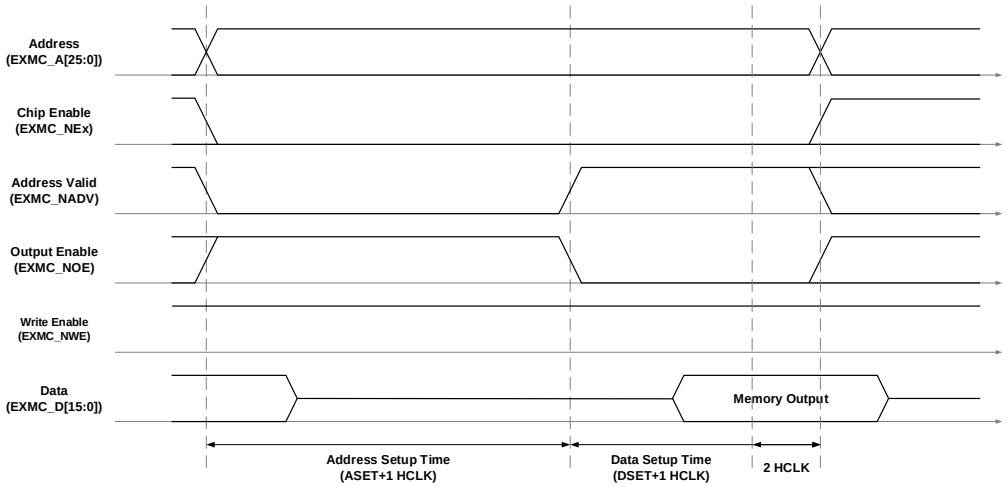
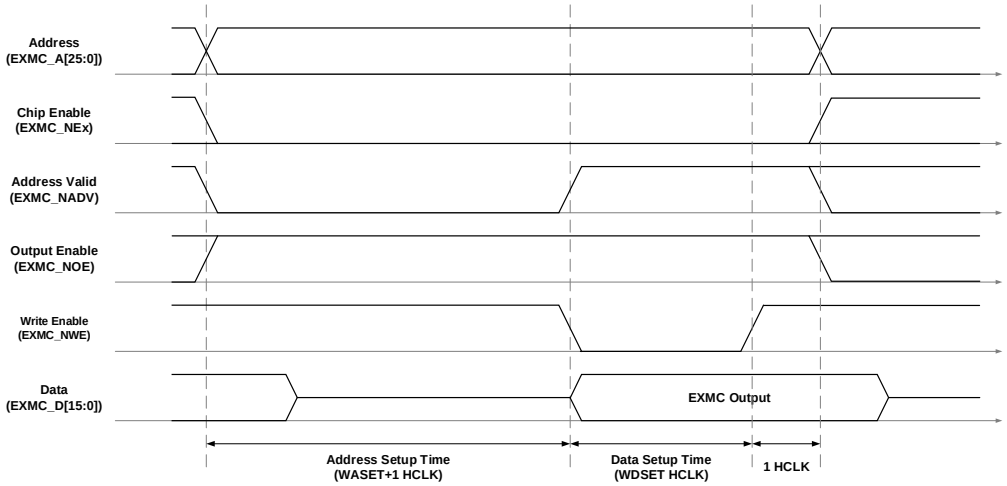


Figure 25-14. Mode C write access



The differences between mode C and mode 1 write timing are 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, and the toggle of NOE and NADV are different.

Table 25-9. Mode C related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWAIT	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

Bit Position	Bit Name	Reference Setting Value
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
27-20	Reserved	0x00
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
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 25-15. Mode D read access

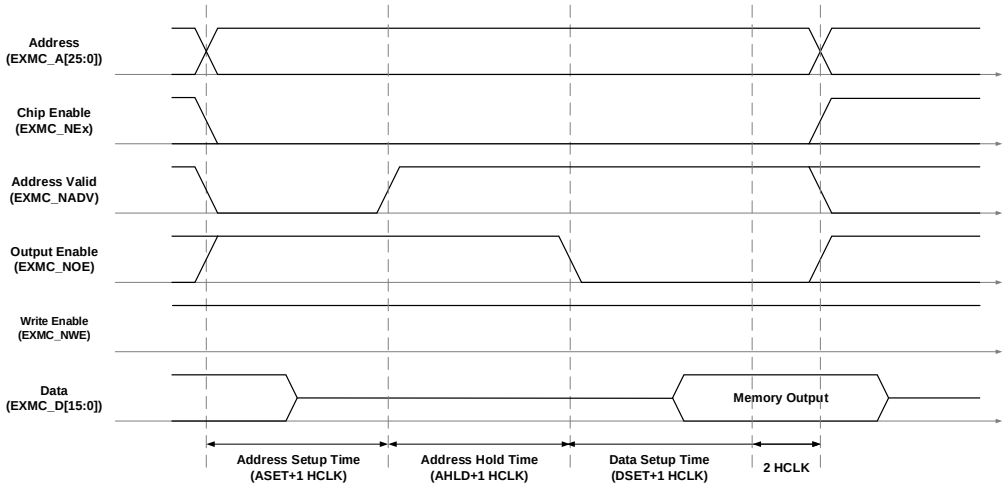


Figure 25-16. Mode D write access

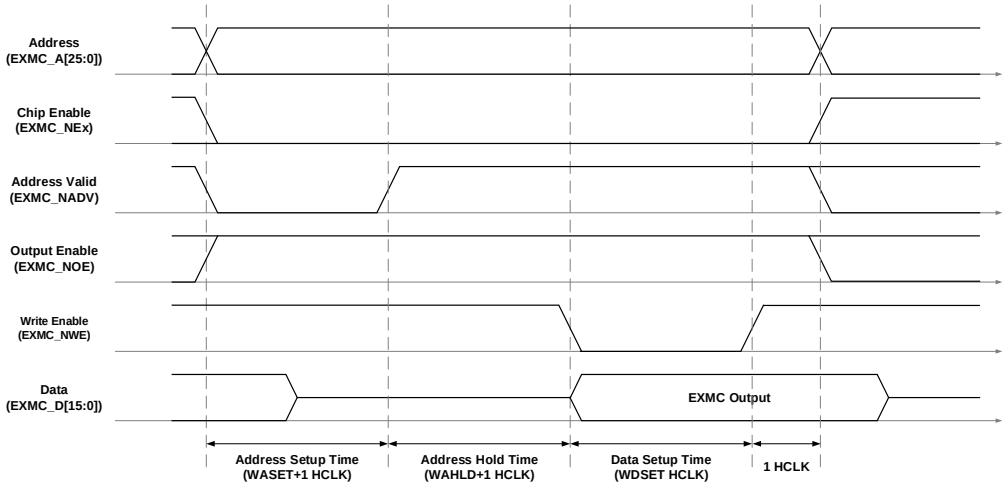


Table 25-10. Mode D related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWAIT	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)

Bit Position	Bit Name	Reference Setting Value
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
27-20	Reserved	0x00
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	WDSET	Depends on memory and user (WDSET+1HCLK for write)
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 25-17. Multiplex mode read access

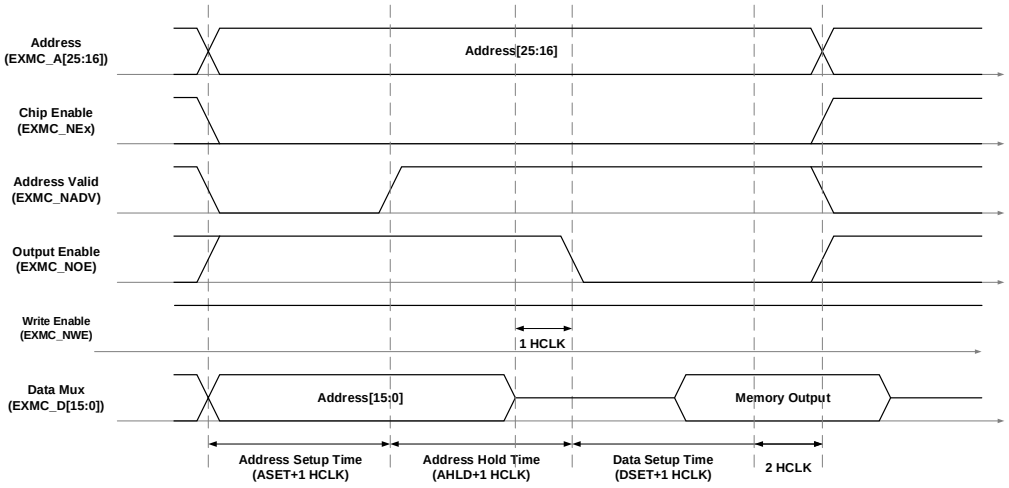


Figure 25-18. Multiplex mode write access

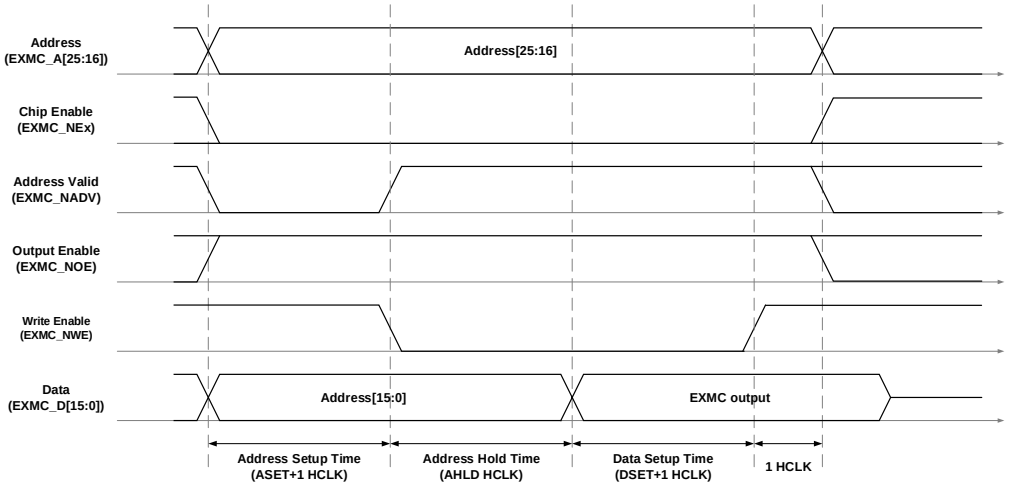


Table 25-11. Multiplex mode related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWAIT	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 (DSET+2 HCLK for write, DSET+3 HCLK for read)
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:

$$T_{DATA\_SETUP} \geq \max T_{WAIT\_ASSERTION} + 4HCLK \quad (25\text{-}1)$$

If memory wait signal is aligned to EXMC_NE:

If

$$maxT_{WAIT\_ASSERTION} \geq T_{ADDRES\_PHASE} + T_{HOLD\_PHASE} \quad (25\text{-}2)$$

be

$$T_{DATA\_SETUP} \geq (maxT_{WAIT\_ASSERTION} - T_{ADDRES\_PHASE} - T_{HOLD\_PHASE}) + 4HCLK \quad (25\text{-}3)$$

Otherwise

$$T_{DATA\_SETUP} \geq 4HCLK \quad (25\text{-}4)$$

Figure 25-19. Read access timing diagram under async-wait signal assertion

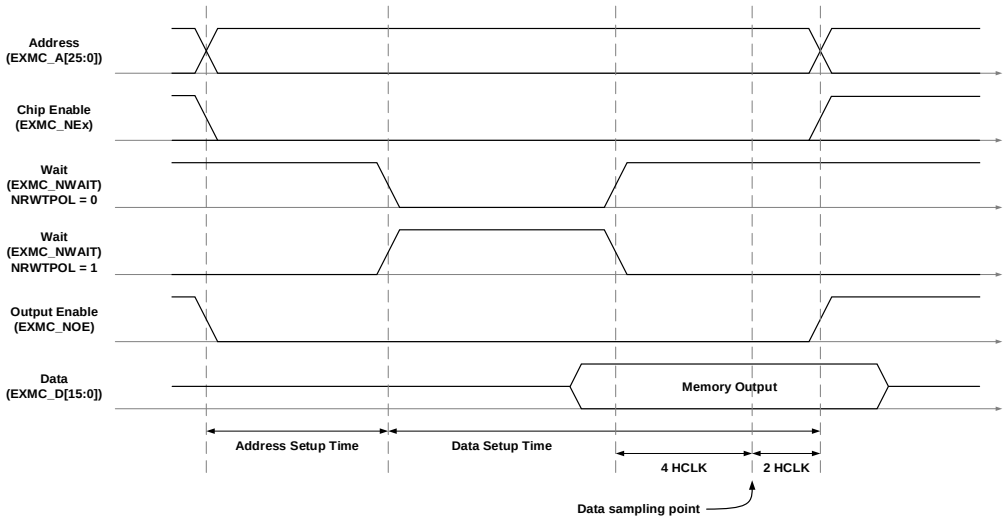
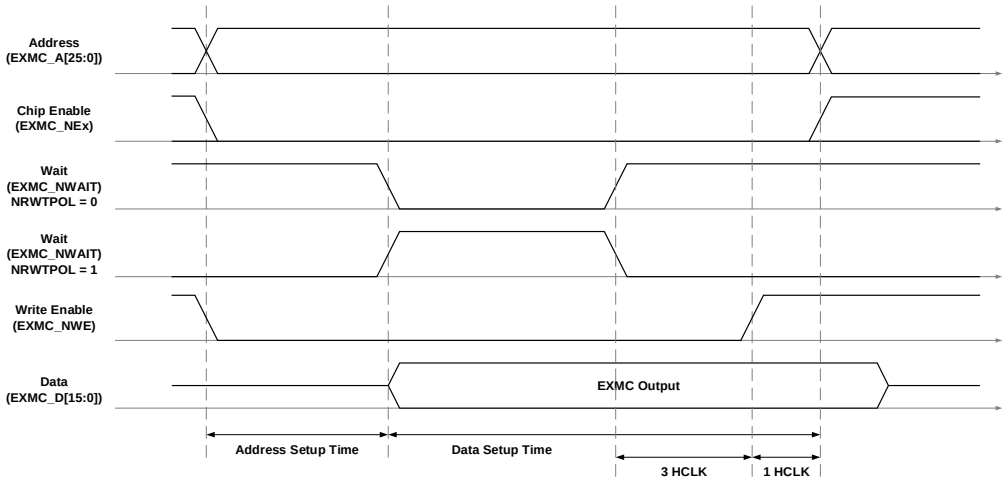


Figure 25-20. Write access timing diagram under async-wait signal assertion



Synchronous access timing diagram

The relations between memory clock (EXMC_CLK) and system clock (HCLK) clock are as follows:

$$EXMC\_CLK = \frac{HCLK}{CKDIV+1} \quad (25\text{-}5)$$

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:

NOR Flash latency=DLAT+2 (25-6)

For NOR Flash's specification including the EXMC_NADV cycle, their relationship should be:

NOR Flash latency=DLAT+3 (25-7)

2. Data wait

Users should guarantee that EXMC_NWAIT signal matches that of the external device. This signal 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 is 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. Automatic burst split at CRAM page boundary

Crossing page boundary burst access is prohibited in CRAM 1.5, an automatic burst split functionality is implemented by the EXMC. To guarantee correct burst split operation, users should specify CRAM page size by configuring the CPS bit in EXMC_SNCTLx register to inform the EXMC when this functionality should be performed.

4. 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 25-3. EXMC bank 0 supports all transactions***.

Synchronous mux burst read timing - NOR, PSRAM (CRAM)

Figure 25-21. Read timing of synchronous multiplexed burst mode

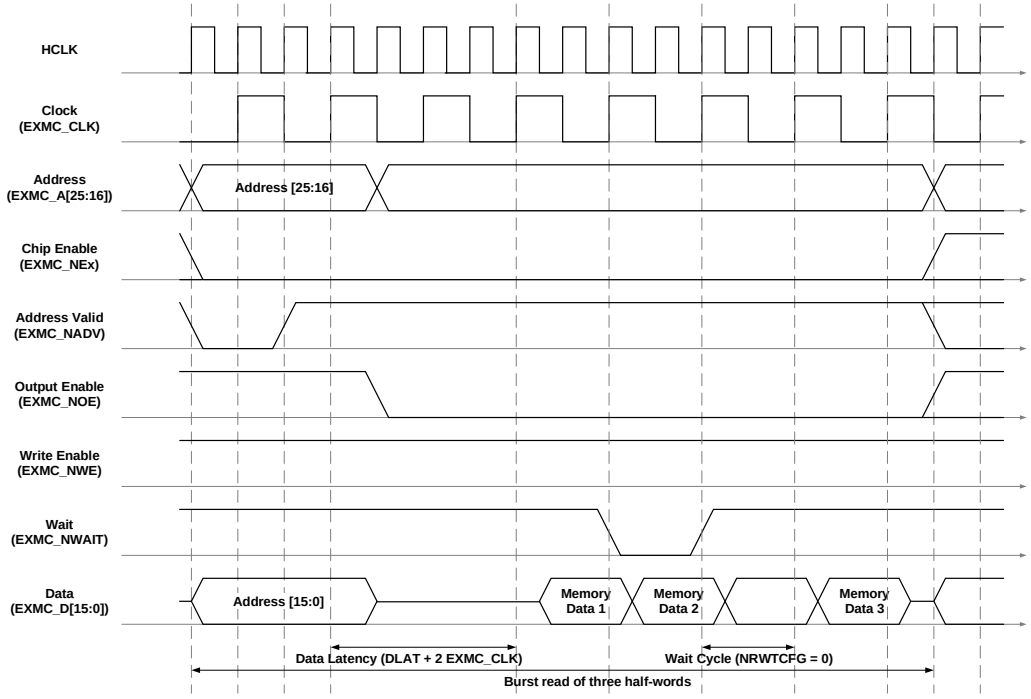


Table 25-12. Timing configurations of synchronous multiplexed read mode

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-20	Reserved	0x000
19	SYNCWR	No effect
18-16	CPS	0x0
15	ASYNCWAIT	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

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
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 25-22. Write timing of synchronous multiplexed burst mode

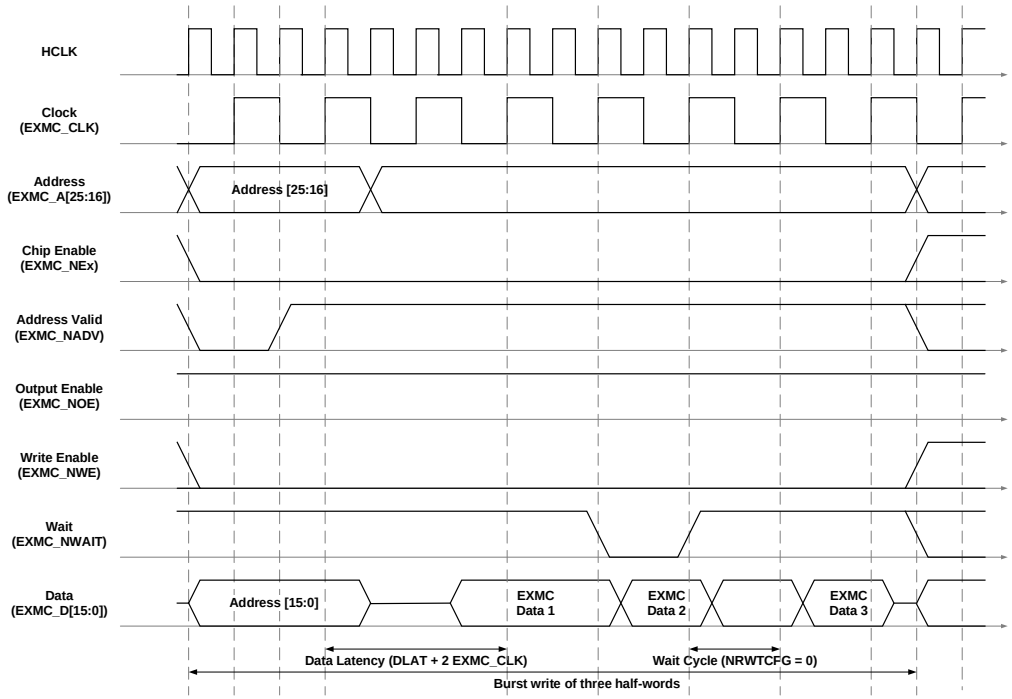


Table 25-13. Timing configurations of synchronous multiplexed write mode

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-20	Reserved	0x000
19	SYNCWR	0x1, synchronous write enable
18-16	CPS	0x0

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
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

25.3.5. 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-bit 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 25-14. 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 Pin	Direction	Functional description
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 25-15. 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_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 25-16. 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,

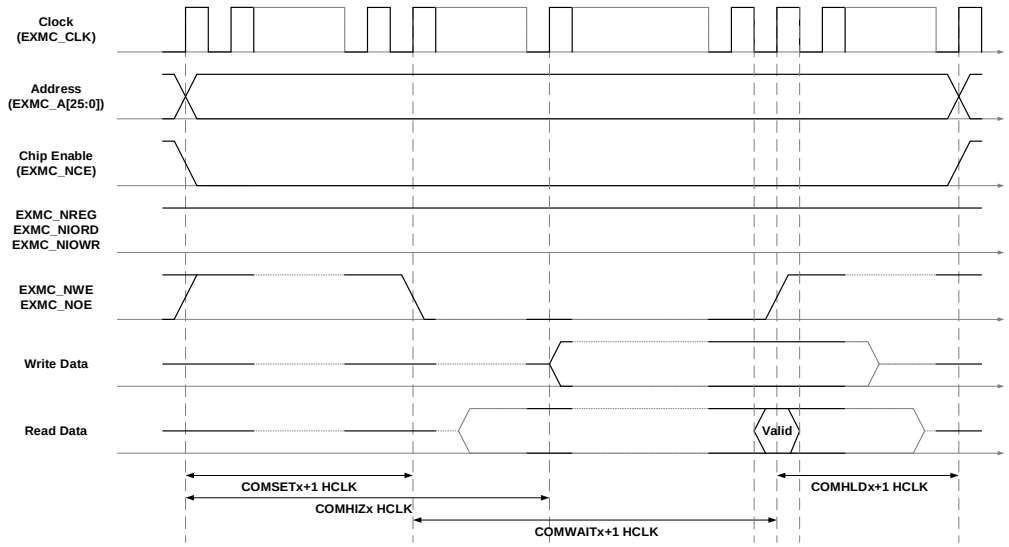
such as EXMC_NPCTLx, EXMC_NPINTENx, EXMC_NPCTCFGx, EXMC_NPATCFGx, EXMC_PIOTCFG3 and EXMC_NECCx. Among these registers, EXMC_NPCTCFGx, EXMC_NPATCFGx, EXMC_PIOTCFG3 registers contain four timing parameters individually which are configured according to user specification and features of the external memory.

Table 25-17. 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	254
Memory wait time (WAIT)	W/R	HCLK	Minimum duration of sending command	2	256
Memory setup time (SET)	W/R	HCLK	The number of HCLK clock cycles to build address before sending command	1	255

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 25-23. 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 (EXMC_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:

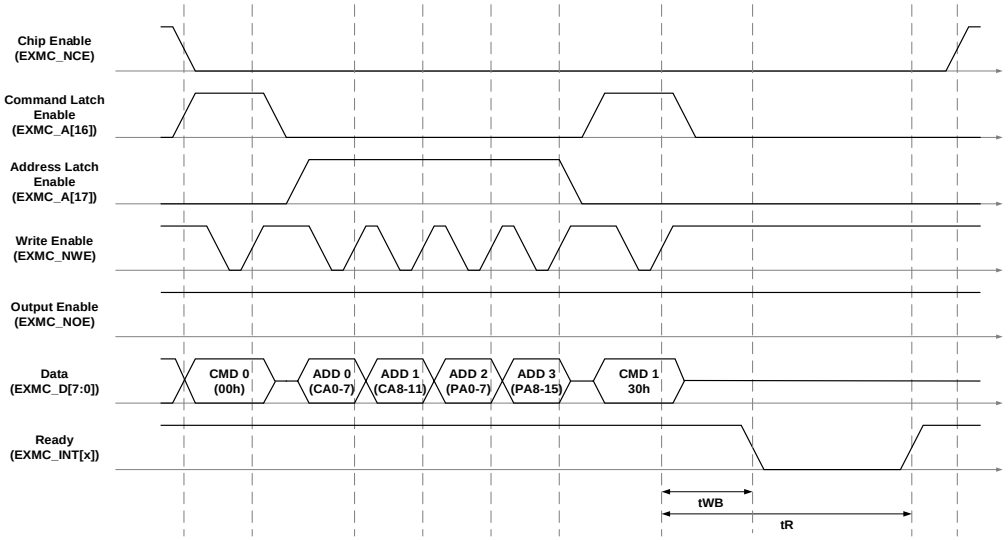
1. Configure EXMC_NPCTLx and EXMC_NPCTCFGx register. When pre-waiting is needed, EXMC_NPATCFGx has to be configured.
2. Send the command of NAND Flash read operation to the common space. Namely, during the valid period of EXMC_NCE and EXMC_NWE, when EXMC_CLE (EXMC_A[16]) becomes valid (high level), data 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 is send, 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 25-24. 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 t_{WB} (t_{WB} 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 t_{WB}. 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 MCU 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 the EXMC_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 1.

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.

25.4. Registers definition

EXMC base address: 0xA000 0000

25.4.1. 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 30DX

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												SYNCWR	CPS[2:0]		
												rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNCWAIT	EXMODEN	NRWTEN	WREN	NRWTCFG	WRAPEN	NRWTPOL	SBRSTEN	Reserved	NREN	NRW[1:0]		NRTP[1:0]		NRMUX	NRBKEN
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	CPS[2:0]	CRAM page size 000: Automatic burst split on page boundary crossing 001: 128 bytes 010: 256 bytes 011: 512 bytes 100: 1024 bytes Others: Reserved
15	ASYNCWAIT	Asynchronous wait 0: Disable the asynchronous wait function 1: Enable the asynchronous wait function
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 NWAIT 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

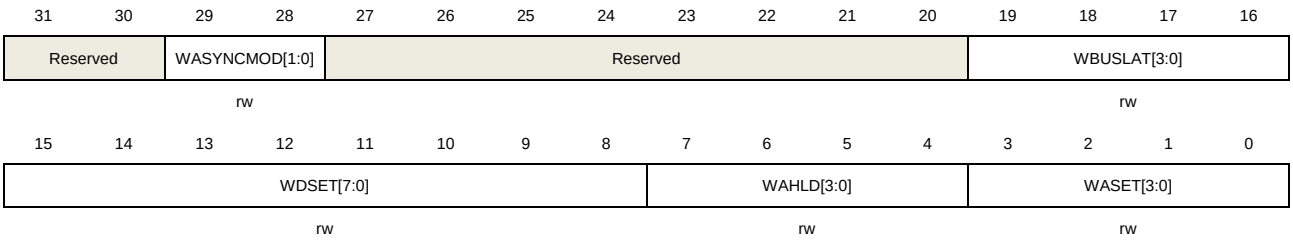
		This field is meaningful only in asynchronous access. 0x00: Reserved 0x01: Data setup time = 2 * HCLK period …… 0xFF: Data setup time = 256 * HCLK period
7:4	AHLD[3:0]	Address hold time This field is used to set the time of address hold phase, which only used in mode D and multiplexed mode. 0x0: Reserved 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.



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
27:20	Reserved	Must be kept at reset value.
19:16	WBUSLAT[3:0]	Bus latency Bus latency added at the end of each write transaction to match with the minimum time between consecutive transactions. 0x0: Bus latency = 1 * HCLK period 0x1: Bus latency = 2 * HCLK period …… 0xF: Bus latency = 16 * HCLK period
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

25.4.2. 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 function enable

		0: Disable wait function 1: Enable wait function
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 0046 (for bank1 and bank2), 0x0000 0040 (for bank3)

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: Reserved
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: Reserved
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: Reserved
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: Reserved

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: Reserved
23:16	ATTHLD[7:0]	Attribute 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: ATTHLD = 1 * HCLK …… 0xFE: ATTHLD = 254 * HCLK

		0xFF: Reserved
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: Reserved
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: Reserved

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 …… 0x00: 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, (x =1, 2)

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(byte)	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]

26. Controller area network (CAN)

26.1. Overview

CAN bus (Controller Area Network) is a bus standard designed to allow microcontrollers and devices to communicate with each other without a host computer.

The CAN network interfaces of the GD32E503xx, GD32E505xx and GD32E507xx series supports the CAN protocol 2.0A and B, and the GD32E508xx series also supports the ISO11898-1:2015 and BOSCH CAN-FD specifications.The CAN interface automatically handles the transmission and the reception of CAN frames. The CAN0 and CAN1 provides 28 scalable/configurable identifier filter banks.The CAN2 independently provides 14 scalable/configurable identifier filter banks. Three transmit mailboxes are provided to the software for transfer messages. The transmission scheduler decides which mailbox will be transmitted firstly. Three complete messages can be stored in every FIFO. The FIFOs are managed completely by hardware. Two receiving FIFOs are used by hardware to store the incoming messages. In addition, the CAN controller provides all hardware functions, which supports the time-triggered communication option, in safety-critical applications.

26.2. Characteristics

- Supports CAN protocols version 2.0A, B
- Classical frames: baud rates up to 1 Mbit/s
- Supports transmitter delay compensation.
- Supports the time-triggered communication
- Interrupt enable and clear
- Supports CAN FD Frame (only in GD32E508xx series)
- FD frames: baud rates up to 6 Mbit/s (only in GD32E508xx series)
- Supports transmitter delay compensation (only in GD32E508xx series)

Transmission

- Supports 3 transmit mailboxes.
- Supports priority of transmission message.
- Supports time stamp at SOF transmission.

Reception

- Supports 2 Rx FIFOs and each has 3 messages depth
- 28 scalable/configurable identifier filter banks in CAN0 and CAN1
- 14 independent and scalable/configurable identifier filter banks in CAN2
- FIFO lock

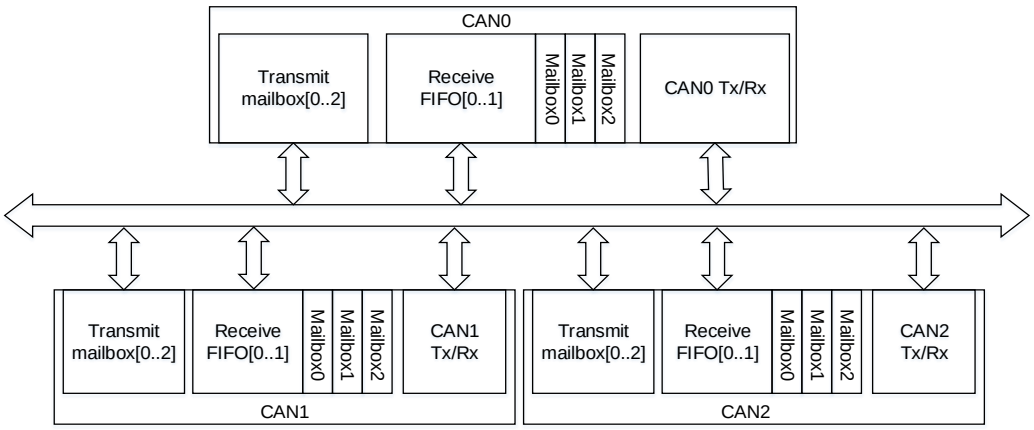
Time-triggered communication

- Disable retransmission automatically in time-triggered communication mode.
- 16-bit free timer
- Time stamp on SOF reception
- Time stamp sent in last two data bytes

26.3. Function overview

Figure 26-1. CAN module block diagram shows the CAN block diagram.

Figure 26-1. CAN module block diagram



26.3.1. 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 and 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 by hardware.

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 configuration 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 could communicate with other CAN communication nodes in normal working mode.

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.

26.3.2. 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 could detect 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 for monitoring the network messages.

Loopback communication mode

Loopback communication mode means the transmitted messages are transferred into the Rx FIFOs, the RX pin is disconnected from the CAN network and the TX pin can still send messages to the CAN network.

Setting LCMOD bit in CAN_BT register to enter loopback communication mode, while clearing it to leave. Loopback communication mode is useful for 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 transmitted messages are transferred into the Rx FIFOs.

Setting LCMOD and SCMOD bit in CAN_BT register to enter loopback and silent communication mode, while clearing them to leave.

Loopback and silent communication mode is used for self-test. The TX pin holds in recessive state. The RX pin holds in high impedance state.

Normal communication mode

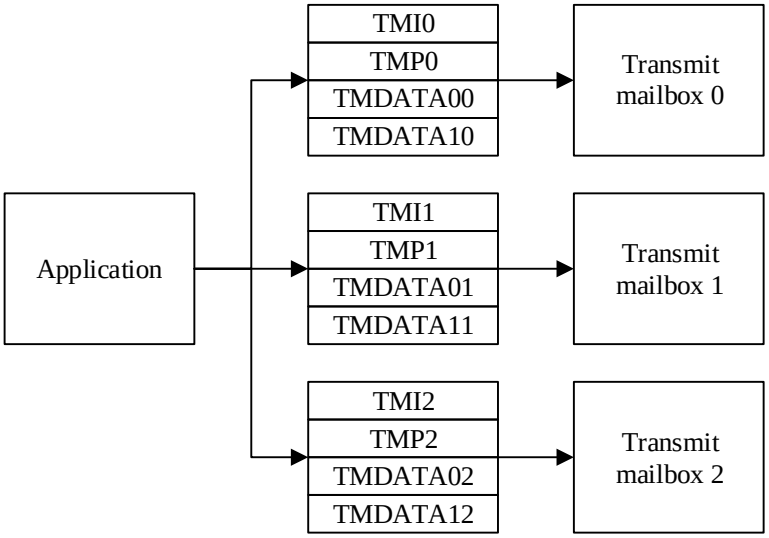
Normal communication mode is the default communication mode when the LCMOD and SCMOD bits in CAN_BT register are cleared.

26.3.3. Data transmission

Transmission register

Three transmit mailboxes are used for the application. Transmit mailboxes are used by configuring four transmission registers: CAN_TMIx, CAN_TMPx, CAN_TMDATA0x and CAN_TMDATA1x. As is shown in ***Figure 26-2. Transmission register***.

Figure 26-2. Transmission register

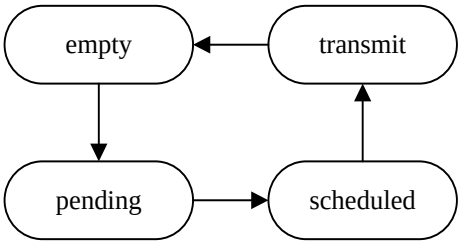


If FD frame would be transmitted, always write TMDATA00 registers when mailbox 0 is used, TMDATA01 register when mailbox 1 is used and TMDATA02 register when mailbox 2 is used until the end. For example, if application wants to transmit 64 bytes data using mailbox0, it needs to write the 64 bytes data through TMDATA00 register for 16 times. The data is stored in internal SRAM.

Transmit mailbox state

A transmit mailbox can be used when it is free (**empty state**). If the mailbox is filled with data, set 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 scheduling the transmission (**scheduled state**). A mailbox with highest priority enters into transmit state and starts transmitting the message (**transmit state**). After the message has been sent, the mailbox is free (**empty state**). As is shown in ***Figure 26-3. State of transmit mailbox***.

Figure 26-3. State of transmit mailbox



Transmit status and error

The CAN_TSTAT register includes the transmit status and error bits: MTF, MTFNERR, MAL, MTE.

- MTF: mailbox transmit finished. Typically, MTF is set when the frame in the transmit mailbox has been sent.
- MTFNERR: mailbox transmit finished with no error. MTFNERR is set when the frame in the transmit mailbox has been sent without any error.
- MAL: mailbox arbitration lost. MAL is set when the frame transmission is failed due to the arbitration lost.
- MTE: mailbox transmit error. MTE is set when the frame transmission is failed due to the error detected on the CAN bus.

Steps of sending a frame

To send a frame through the CAN:

Step 1: Select one free transmit mailbox.

Step 2: Configure four transmission registers with the application's acquirement.

Step 3: Set TEN 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 transmit mailbox's status is **pending** or **scheduled**, the abort of transmission can be done immediately.

In the **transmit** state, the abort of transmission does not take effect immediately until the transmission is finished. In case that the transmission is successful, the MTFNERR and MTF in CAN_TSTAT are set and state changes to be **empty**. In case that the transmission is 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 that TFO is 1, the three transmit mailboxes work first-in first-out (FIFO).

In case that 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 firstly.

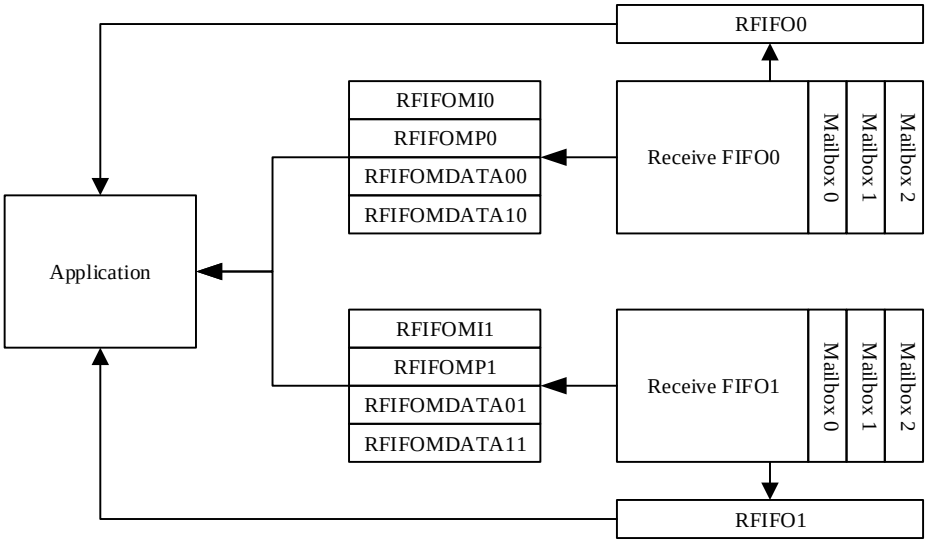
26.3.4. Data reception

Reception register

Two Rx FIFOs are used for the application. Rx FIFOs are managed by 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 is shown in ***Figure 26-4. Reception register***.

Figure 26-4. Reception register



Rx FIFO

Rx FIFO has three mailboxes. The reception frames are stored in the mailbox according to the arriving sequence. First arrived frame can be accessed by application firstly.

The number of frames in the Rx 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 Rx FIFO0, the frame data is stored in the CAN_RFIFOMI0, CAN_RFIFOMP0, CAN_RFIFOMDATA00 and CAN_RFIFOMDATA10 registers. After reading the current frame, set RFD bit in CAN_RFIFO0 to release a frame in the Rx FIFO and the software can read the next frame.

If FD frame has been received, the data always read from CAN_RFIFOMDATA00 register for FIFO0 and CAN_RFIFOMDATA01 for FIFO1 until the end. For example, if application needs to read 64 bytes data from FIFO0. It needs to read the 64 bytes data through CAN_RFIFOMDATA00 register for 16 times. The received data is stored in internal SRAM.

Rx FIFO status

RFL (Rx FIFO length) bits in CAN_RFIFOx register is 0 when no frame is stored in the Rx FIFO and it is 3 when FIFOx is full.

When RFF bit in CAN_RFIFOx register is set, it indicates FIFOx is full, at this time, RFL is 3.

When a new frame arrives after the FIFO has held three frames, the RFO bit in CAN_RFIFOx register will be set, and it indicates FIFOx is overrun. 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 Rx FIFO and the last frame in the Rx FIFO is discarded.

Steps of receiving a message

Step 1: Check the number of frames in the Rx FIFO.

Step 2: Read CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_RFIFOMDATA1x.

Step 3: Set the RFD bit in CAN_RFIFOx register.

26.3.5. Filtering function

The CAN receives frames from the CAN bus. If the frame passes the filter, it is stored in the Rx FIFOs. Otherwise, the frame will be discarded without intervention by the software.

The identifier of frame is used for the matching of the filter.

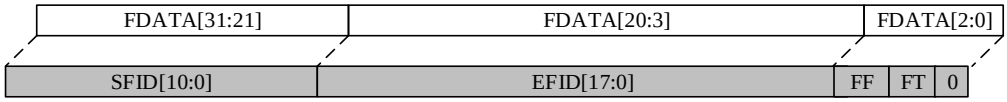
Scale

CAN0 and CAN1 has the filter consists of 28 banks: bank0 to bank27. CAN2 has independent the filter consists of 14 banks: bank0 to bank13. Each bank has two 32-bit registers: CAN_FxDATA0 and CAN_FxDATA1.

Each filter bank can be configured to 32-bit or 16-bit.

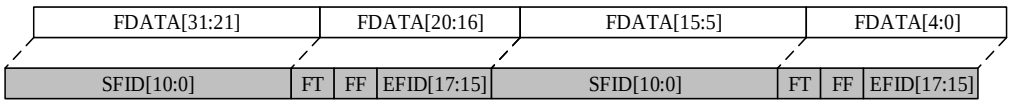
32-bit: SFID[10:0], EFID[17:0], FF and FT bits. As is shown in ***Figure 26-5. 32-bit filter***.

Figure 26-5. 32-bit filter



16-bit: SFID [10:0], FT, FF and EFID[17:15] bits. As is shown in ***Figure 26-6. 16-bit filter***.

Figure 26-6. 16-bit filter



Mask mode

For the Identifier of a data frame to be filtered, the mask mode is used to specify which bits must be the same as the preset Identifier and which bits need not be judged. 32-bit mask mode example is shown in ***Figure 26-7. 32-bit mask mode filter***.

Figure 26-7. 32-bit mask mode filter

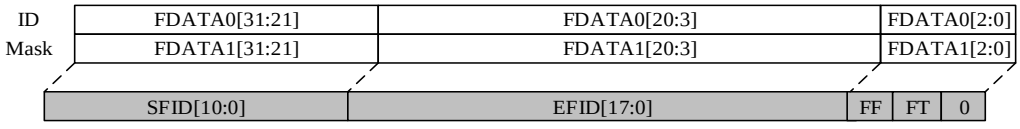
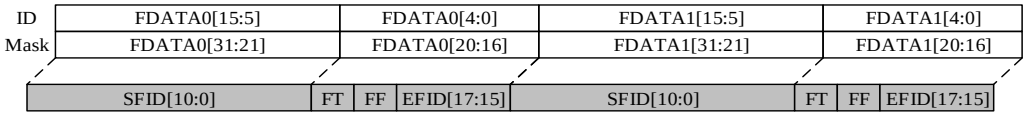


Figure 26-8. 16-bit mask mode filter



List mode

The filter consists of frame identifiers. The filter can determine 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 26-9. 32-bit list mode filter***.

Figure 26-9. 32-bit list mode filter

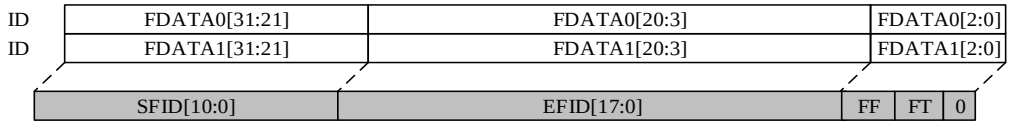
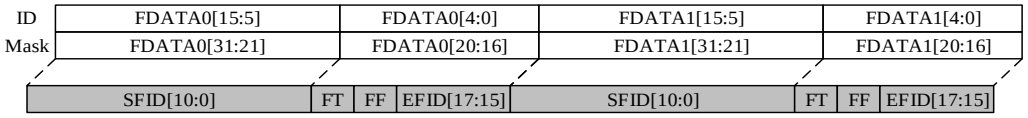


Figure 26-10. 16-bit list mode filter



Filter number

Filter consists of some filter bank. According to the mode and the scale of each of the filter banks, filter has different effects.

For example, there are two filter banks. Bank0 is configured as 32-bit mask mode. Bank1 is configured as 32-bit list mode. The filter number is shown in ***Table 26-1. 32-bit filter number***.

Table 26-1. 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 the bank will be stored in the FIFO0.

Active

The filter bank needs to be activated if the bank is to be used, otherwise, the filter bank should be left deactivated.

Filtering index

Each filter number corresponds to a filtering rule. When the frame which is associated with a filter number N passes the filters, the filter index is N. It stores in the FI bits in CAN_RFIFOMPx.

Filter bank has filter index once it is associated with the FIFO no matter whether the bank is active or not.

The example about filtering index is shown in ***Table 26-2. Filtering index***.

Table 26-2. Filtering index

Filter bank	FIFO0	Active	Filter nunber	Filter bank	FIFO1	Active	Filter nunber
0	F0DATA0-32bits-ID	Yes	0	2	F2DATA0[15:0]-16bits-ID	Yes	0
	F0DATA1-32bits-Mask				F2DATA0[31:16]-16bits-Mask		
1	F1DATA0-32bits-ID	Yes	1		F2DATA1[15:0]-16bits-ID		1
	F1DATA1-32bits-ID		2		F2DATA1[31:16]-16bits-Mask		
3	F3DATA0[15:0]-16bits-ID	No	3	4	F4DATA0-32bits-ID	No	2
	F3DATA0[31:16]-16bits-Mask				F4DATA1-32bits-Mask		
	F3DATA1[15:0]-16bits-ID		4	5	F5DATA0-32bits-ID	No	3
	F3DATA1[31:16]-16bits-Mask				F5DATA1-32bits-ID		4
7	F7DATA0[15:0]-16bits-ID	No	5	6	F6DATA0[15:0]-16bits-ID	Yes	5
	F7DATA0[31:16]-16bits-ID		6		F6DATA0[31:16]-16bits-ID		6
	F7DATA1[15:0]-16bits-ID		7		F6DATA1[15:0]-16bits-ID		7
	F7DATA1[31:16]-16bits-ID		8		F6DATA1[31:16]-16bits-ID		8
8	F8DATA0[15:0]-16bits-ID	Yes	9	10	F10DATA0[15:0]-16bits-ID	No	9
	F8DATA0[31:16]-16bits-ID		10		F10DATA0[31:16]-16bits-Mask		
	F8DATA1[15:0]-16bits-ID		11		F10DATA1[15:0]-16bits-ID		10
	F8DATA1[31:16]-16bits-ID		12		F10DATA1[31:16]-16bits-Mask		
9	F9DATA0[15:0]-16bits-ID	Yes	13	11	F11DATA0[15:0]-16bits-ID	No	11

Filter bank	FIFO0	Active	Filter nunber	Filter bank	FIFO1	Active	Filter nunber
	F9DATA0[31:16]-16bits-Mask				F11DATA0[31:16]-16bits-ID		12
	F9DATA1[15:0]-16bits-ID		14		F11DATA1[15:0]-16bits-ID		13
	F9DATA1[31:16]-16bits-Mask				F11DATA1[31:16]-16bits-ID		14
12	F12DATA0-32bits-ID	Yes	15	13	F13DATA0-32bits-ID	Yes	15
	F12DATA1-32bits-Mask				F13DATA1-32bits- ID		16

Priority

The filters have the priority rules:

32-bits mode is higher than 16-bits mode.

List mode is higher than mask mode.

Smaller filter number has the higher priority.

26.3.6. 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 time points of message transmission are pre-defined.

In this mode, an internal 16-bit counter starts working, incrementing by 1 at each CAN bit time. This internal counter provides time stamps for sending and receiving data, stored in registers CAN _RFIFOMPx and CAN_TMPx.

The automatic retransmission is disabled in the time-triggered CAN communication.

26.3.7. Communication parameters

Automatic retransmission forbid mode

In time-triggered communication mode, the requirement for automatic retransmission must be disabled and CAN be met by setting ARD position 1 of the CAN_CTL register.

In this mode, the data is sent only once, and if the transmission fails due to arbitration failure or bus error, the CAN bus controller does not automatically resend the data as usual.

At the end of sending, the MTF bit of register CAN_TSTAT is hardware set to 1, and the sending status information can be obtained via MTFNERR, MAL, and MTE.

Bit time

On the bit-level, the CAN protocol uses synchronous bit transmission. This not only enhances the transmitting capacity but also requires a sophisticated method of bit synchronization. While bit synchronization in a character-oriented transmission (asynchronous) is performed upon the reception which the start bit is available with each character, the synchronous transmission protocol just need one start bit available at the beginning of a frame. To ensure that the receiver correctly reads the messages, resynchronization is required. Phase buffer segments' sample point of the front-end and back-end should be inserted a bit interval.

The CAN protocol regulates bus access by bit-wise arbitration. The signal propagated from sender to receiver and back to the sender must be completed within one bit-time. For synchronization, in addition to the phase buffer segments, a propagation delay segment is needed. The propagation delay segment is regarded as signal delays caused by transmitting and receiving nodes in the process of the signal propagation on the bus.

The normal bit time from the CAN protocol has three segments as follows:

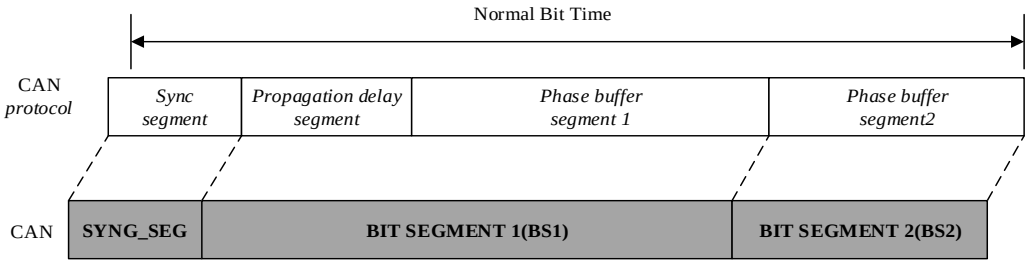
Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ($1 \times t_{CAN}$).

Bit segment 1 (BS1): It defines the location of the sample point. It includes the Propagation delay segment and Phase buffer segment 1 in the CAN standard. Its duration is programmable from 1 to 16 time quanta but it may be automatically lengthened to compensate for positive phase drifts due to different frequency of the various nodes of the network.

Bit segment 2 (BS2): It defines the location of the transmit point. It represents the Phase buffer segment 2 in the CAN standard. Its duration is programmable from 1 to 8 time quanta but it may also be automatically shortened to compensate for negative phase drifts.

The bit time is shown as in the ***Figure 26-11. The bit time***.

Figure 26-11. The bit time



The resynchronization Jump Width (SJW): it can be lengthened or shortened to compensate for the Synchronization error of the CAN network node. It is programmable from 1 to 4 time quanta.

A valid edge is defined as the first toggle in a bit time from dominant to recessive bus level before the controller sends a recessive bit.

If a valid edge is detected in BS1, not in SYNC_SEG, BS1 is added up to SJW maximumly, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2, not in SYNC_SEG, BS2 is cut down to SJW at most, so that the transmit point is moved earlier.

Baud rate

The clock of the CAN derives from the APB1 bus. The CAN calculates its baud rate as follow:

$$\text{BaudRate}=\frac{1}{\text{Normal Bit Time}} \quad (22\text{-}1)$$

$$\text{Normal Bit Time}=t_{SYNC\_SEG}+t_{BS1}+t_{BS2} \quad (22\text{-}2)$$

with:

$$t_{SYNC\_SEG}=1\times t_q \quad (22\text{-}3)$$

$$t_{BS1}=(1+\text{BT.BS1})\times t_q \quad (22\text{-}4)$$

$$t_{BS2}=(1+\text{BT.BS2})\times t_q \quad (22\text{-}5)$$

$$t_q=(1+\text{BT.BAUDPSC})\times t_{PCLK1} \quad (22\text{-}6)$$

26.3.8. CAN FD operation

The CAN FD function is enabled by setting FDEN to 1 in CAN_FDCTL register. If FDEN bit is cleared, only calassical frames are supported. If FDEN bit is set, it supports calassical frames and FD frames. Whether the current frame is FD or not could be defined by received FDF bit (the previously reserved bit in CAN frames with 11-bit identifiers or the first previously reserved bit in CAN frames with 29-bit identifiers which now is decoded as FDF bit). If FDF bit is recessive, meaning to be the CAN FD frame, otherwise FDF bit is dominant, meaning to be the classical frame.

The CAN FD supports ISO11898-1 or Bosch CAN FD Specification V1.0 by configuring NISO bit in CAN_FDCTL register.

The two bits following the FDF bit are reserved bit and BRS bit respectively. The received BRS bit determines the bit rate of data. When BRS is dominant, bit rate of data could not switch by configuring the CAN_DBT register. When BRS is recessive, bit rate of data could switch to a higher bit rate inside the frame by configuring the CAN_DBT register. The bit rate of data can be switched in the period from BRS bit to CRC delimiter (refer to ISO11898-1 or Bosch CAN FD Specification V1.0).

When Protocol Exception Handling is enabled (PRED bit in CAN_FDCTL register is 0), it causes the operation state to change to be IDLE and interrupts current frame at the next sample point when recessive reserve bit is received. When Protocol Exception Handling is disabled (PRED bit in CAN_FDCTL register is 1), it will treat a recessive reserve bit as a form error and respond with an error frame. If any recessive reserve bit occurs, set PRE bit in

CAN_FDSTAT register to 1.

The transmission of ESI bit (the bit before DLC bits, refer to ISO11898-1 or Bosch CAN FD Specification V1.0) is defined by ESIMOD bit in CAN_FDCTL register and ESI bit in CAN_TMPx register. If ESIMOD bit is 0, it will transmit the dominant bit by error active nodes and transmit the recessive bit by error passive nodes. If ESIMOD bit is set, it will transmit ESI bit in CAN_TMPx register.

The transmission of FDF bit and BRS bit is defined by FDF bit and BRS bit in CAN_TMPx registers.

26.3.9. Transmitter Delay Compensation

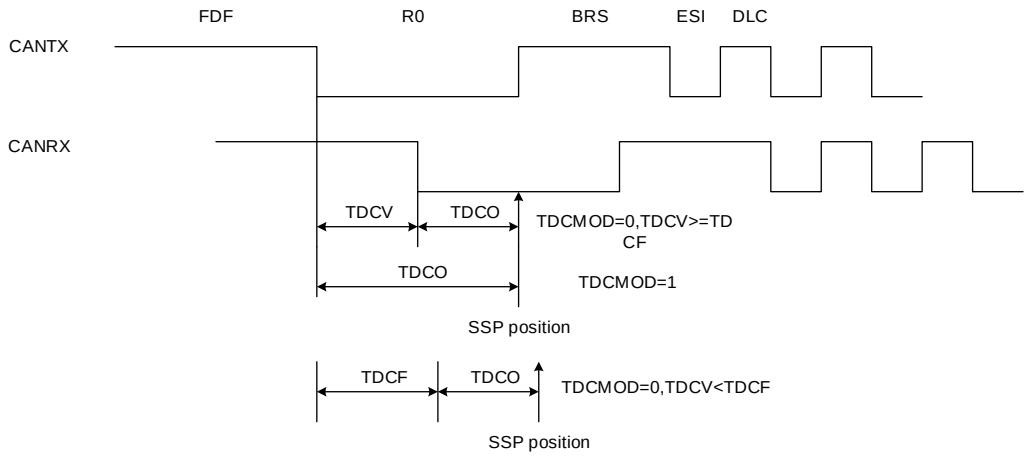
CAN FD supports transmitter delay compensation mechanism, it could be used in applications when the length of the CAN bit time in the DATA-PHASE is shorter than the limit required by the transceiver's internal delay time. The description of transmitter delay compensation, please refer to ISO11898-1 or Bosch CAN FD Specification V1.0.

The transmitter delay compensation is enabled by setting TDCEN bit in CAN_FDCTL register.

The transmitter delay compensation is used to adjust the position of the SSP. The SSP delay is defined as the delay from dominant edges on CANTX to SSP point. If TDCMOD bit in CAN_FDCTL register is set, the SSP delay is defined in TDCO bits of the CAN_FDTDC registers by software. If TDCMOD bit in CAN_FDCTL register is cleared, the hardware automatically uses the falling edges between FDF bit and RES0 bit to calculate the delay from dominant edges on CANTX to dominant edges on CANRX, and store the delay in TDCV bits of CAN_FDSTAT registers. In case that there is dominant glitch, SSP position would be advanced than expected, leading to a calculated error in compensation measurement. To solve this problem, TDCF bits of CAN_FDTDC register could be used to avoid too small TDCV bits. So the value of SSP delay is TDCO bits add TDCV bits if TDCV is larger than TDCF, or else the value of SSP delay is TDCO bits add TDCF bits.

The SSP delay can not exceed 3 data bit time.

Figure 26-12. Transmitter Delay Measurement



26.3.10. Error flags

The state of CAN bus can be reflected by Transmit Error Counter (TECNT) and Receive Error Counter (RECNT) of CAN_ERR register. The value can be increased or decreased by the hardware according to the error, and the software can judge the stability of the CAN network by these values. For details on incorrect counting, refer to the CAN protocol section.

By using the CAN_INTEN register (ERRIE bit, etc.), the software can control the interrupt generation when error is detected.

Bus-Off recovery

The CAN controller is in Bus-Off state when TECNT is over than 255. In This state, BOERR bit is set in CAN_ERR register, and no longer able to transmit and receive messages.

According to the ABOR configuration in register CAN_CTL, there are two ways to recover from Bus-Off (to an Error active state). Both of these methods require the CAN bus controller in the Bus-Off state to detect the Bus-Off recovery sequence defined by CAN protocol (when CAN_RX detects 128 consecutive 11-bit recessive bits) before automatic recovery.

If ABOR is set, it will be automatically recovered when a Bus-Off recovery sequence is detected.

If ABOR is cleared, CAN controller must be configured to enter initialization mode by setting IWMOD bit in CAN_CTL register, then exit and enter nomal mode. After this operation, it will recover when the recovering sequence is detected.

26.3.11. CAN interrupts

The CAN bus controller occupies 4 interrupt vectors, which are controlled by the register CAN_INTEN.

The interrupt sources can be classified as:

- 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 Rx FIFO0 interrupt can be generated by the following conditions:

- Rx FIFO0 not empty: RFL0 bits in the CAN_RFIFO0 register are not ‘00’ and RFNEIE0 in CAN_INTEN register is set.
- Rx FIFO0 full: RFF0 bit in the CAN_RFIFO0 register is set and RFFIE0 in CAN_INTEN register is set.
- Rx FIFO0 overrun: RFO0 bit in the CAN_RFIFO0 register is set and RFOIE0 in CAN_INTEN register is set.

Rx FIFO1 interrupt

The Rx FIFO1 interrupt can be generated by the following conditions:

- Rx FIFO1 not empty: RFL1 bits in the CAN_RFIFO1 register are not ‘00’ and RFNEIE1 in CAN_INTEN register is set.
- Rx FIFO1 full: RFF1 bit in the CAN_RFIFO1 register is set and RFFIE1 in CAN_INTEN register is set.
- Rx FIFO1 overrun: RFO1 bit in the CAN_RFIFO1 register is set and RFOIE1 in CAN_INTEN register is set.

Error and working mode change (EWMC) 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.

The CAN bus controller interrupt conditions can refer to ***Table 26-3. CAN Event / Interrupt flags***.

Table 26-3. CAN Event / Interrupt flags

Interrupt event	Interrupt / Event flag		Enable control bit	
Transmit interrupt	Mailbox 0 transmit finished flag (MTF0)		TMEIE	
	Mailbox 1 transmit finished flag (MTF1)			
	Mailbox 2 transmit finished flag (MTF2)			
FIFO0 interrupt	Rx FIFO0 length (RFL0[1:0])		RFNEIE0	
	Rx FIFO0 full (RFF0)		RFFIE0	
	Rx FIFO0 overfull (RFO0)		RFOIE0	
FIFO1 interrupt	Rx FIFO1 length (RFL0[1:0])		RFNEIE1	
	Rx FIFO1 full (RFF0)		RFFIE1	
	Rx FIFO1 overfull (RFO0)		RFOIE1	
EWMC interrupt	Warning error (WERR)	Error interrupt flag (ERRIF)	WERRIE	ERRIE
	Passive error (PERR)		PERRIE	
	Bus-Off error (BOERR)		BOIE	
	Error number (1<= ERRN[2:0] <= 6)		ERRNIE	
	Status change interrupt flag of waking up from sleep working mode (WUIF)		WIE	
	Status change interrupt flag of entering sleep working mode (SLPIF)		SLPWIE	

26.4. CAN registers

CAN0 base address: 0x4000 6400

CAN1 base address: 0x4000 6800

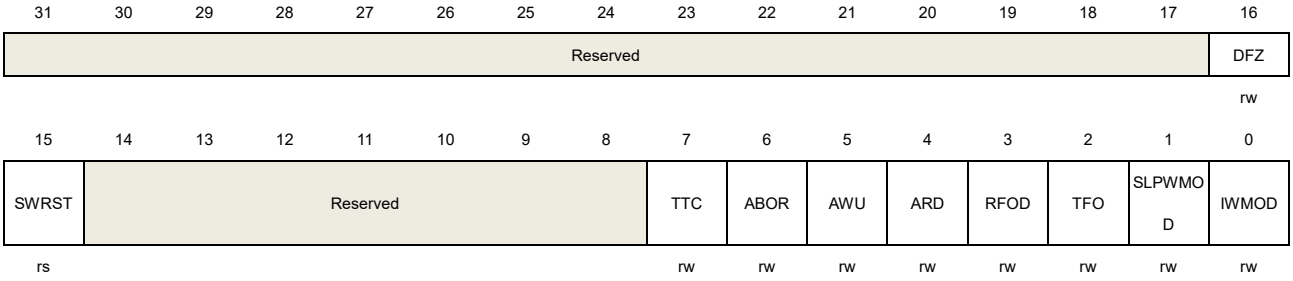
CAN2 base address: 0x4000 CC00

26.4.1. Control register (CAN_CTL)

Address offset: 0x00

Reset value: 0x0001 0002

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



Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value.
16	DFZ	Debug freeze If the CANx_HOLD in DBG_CTL0 register is set, this bit defines the CAN controller is in debug freezing mode or normal working mode. If the CANx_HOLD in DBG_CTL0 register is cleared, this bit takes no effect. 0: CAN reception and transmission work normal even during debug 1: CAN reception and transmission stop working during debug
15	SWRST	Software reset 0: No effect 1: Reset CAN to enter sleep working mode. 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 CAN leaves sleep working mode when CAN bus activity is 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	Rx FIFO overwrite disable 0: Enable Rx FIFO overwrite when Rx FIFO is full and overwrite the FIFO with the incoming frame 1: Disable Rx FIFO overwrite when Rx FIFO is full and discard the incoming frame
2	TFO	Tx FIFO order 0: Order with the identifier of the frame (the smaller identifier has higher priority) 1: Order with first-in and first-out
1	SLPWMOD	Sleep working mode If this bit is set by software, the CAN enters sleep working mode after current transmission or reception is completed. This bit can be cleared by software or hardware. If AWU bit in CAN_CTL register is set, this bit is cleared by hardware when CAN bus activity is 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

26.4.2. 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 entering sleep working mode This bit is set by hardware when entering sleep working mode, and cleared by hardware when the CAN is not in sleep working mode. This bit can also be cleared by software when writting 1 to this bit. 0: CAN is not in the sleep working mode 1: CAN is in the sleep working mode
3	WUIF	Status change interrupt flag of waking up from sleep working mode This bit is set when CAN bus activity event is detected in sleep working mode. This bit can be cleared by software when writting 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 the following events. 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 writting 1 to this bit. 0: No error interrupt event 1: Any error interrupt event has happened
1	SLPWS	Sleep working state This bit is set by hardware when the CAN enters sleep working mode after setting SLPWMOD bit in CAN_CTL register. If the CAN leaves normal working mode to sleep working mode, it must wait the current frame transmission or reception to be completed. This bit is cleared by hardware when the CAN leaves 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 leaving sleep working mode to normal working mode, this bit will be cleared after receiving 11 consecutive recessive bits from the CAN bus. 0: CAN is not in the state of sleep working mode

		1: CAN is in the state of sleep working mode
0	IWS	Initial working state This bit is set by hardware when the CAN enters initial working mode after setting IWMOD bit in CAN_CTL register. If the CAN leaves normal working mode to initial working mode, it must wait the current frame transmission or reception to be completed. This bit is cleared by hardware when the CAN leaves initial working mode after clearing IWMOD bit in CAN_CTL register. If leaving initial working mode to normal working mode, this bit will be cleared after receiving 11 consecutive recessive bits from the CAN bus. 0: CAN is not in the state of initial working mode 1: CAN is in the state of initial working mode

26.4.3. 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 Tx FIFO This bit is set by hardware when transmit mailbox 2 has the last sending order in the Tx FIFO with at least two frames pending.
30	TMLS1	Transmit mailbox 1 last sending in Tx FIFO This bit is set by hardware when transmit mailbox 1 has the last sending order in the Tx FIFO with at least two frames pending.
29	TMLS0	Transmit mailbox 0 last sending in Tx FIFO This bit is set by hardware when transmit mailbox 0 has the last sending order in the Tx FIFO with at least two frames 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 Tx FIFO mailbox in which the frame will be transmitted if at least one mailbox is empty. These bits are the number of the Tx FIFO mailbox in which the frame will be transmitted at last 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 when the transmit error occurs. This bit is reset by writing 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
18	MAL2	Mailbox 2 arbitration lost This bit is set when the arbitration lost occurs. This bit is reset by writting 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
17	MTFNERR2	Mailbox 2 transmit finished with no error This bit is set when the transmission finishes and no error occurs. This bit is reset by writting 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit is reset by hardware when the transmission finishes with error. 0: Mailbox 2 transmit finished with error 1: Mailbox 2 transmit finished with no error
16	MTF2	Mailbox 2 transmit finished This bit is set by hardware when the transmission finishes or aborts. This bit is reset by writting 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 software to stop mailbox 1 transmitting. This bit is reset by hardware when 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 when the transmit error occurs. This bit is reset by writting

		1 to this bit or MTF1 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
10	MAL1	Mailbox 1 arbitration lost This bit is set when the arbitration lost occurs. This bit is reset by writting 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
9	MTFNERR1	Mailbox 1 transmit finished with no error This bit is set when the transmission finishes and no error occurs. This bit is reset by writting 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit is reset by hardware when the transmission finishes with error. 0: Mailbox 1 transmit finished with error 1: Mailbox 1 transmit finished with no error
8	MTF1	Mailbox 1 transmit finished This bit is set by hardware when the transmission finishes or aborts. This bit is reset by writting 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 when 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 when the transmit error occurs. This bit is reset by writting 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
2	MAL0	Mailbox 0 arbitration lost This bit is set when the arbitration lost occurs. This bit is reset by writting 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
1	MTFNERR0	Mailbox 0 transmit finished with no error This bit is set when the transmission finishes and no error occurs. This bit is reset by writting 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit is reset by hardware when the transmission finishes with error. 0: Mailbox 0 transmit finished with error 1: Mailbox 0 transmit finished with no error
0	MTF0	Mailbox 0 transmit finished This bit is set by hardware when the transmission finishes or aborts. This bit is reset by writting 1 to this bit or TEN bit in CAN_TMI0 is 1. 0: Mailbox 0 transmit is progressing

1: Mailbox 0 transmit finished

26.4.4. 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	Rx FIFO0 dequeue This bit is set by software to start dequeuing a frame from Rx FIFO0. This bit is reset by hardware when the dequeuing is done.
4	RFO0	Rx FIFO0 overfull This bit is set by hardware when Rx FIFO0 is overfull and reset by software when writting 1 to this bit. 0: The Rx FIFO0 is not overfull 1: The Rx FIFO0 is overfull
3	RFF0	Rx FIFO0 full This bit is set by hardware when Rx FIFO0 is full and reset by software when writting 1 to this bit. 0: The Rx FIFO0 is not full 1: The Rx FIFO0 is full
2	Reserved	Must be kept at reset value.
1:0	RFL0[1:0]	Rx FIFO0 length These bits are the length of the Rx FIFO0.

26.4.5. 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_w1	rc_w1		r	

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5	RFD1	Rx FIFO1 dequeue This bit is set by software to start dequeuing a frame from Rx FIFO1. This bit is reset by hardware when the dequeuing is done.
4	RFO1	Rx FIFO1 overfull This bit is set by hardware when Rx FIFO1 is overfull and reset by writting 1 to this bit. 0: The Rx FIFO1 is not overfull 1: The Rx FIFO1 is overfull
3	RFF1	Rx FIFO1 full This bit is set by hardware when Rx FIFO1 is full and reset by writting 1 to this bit. 0: The Rx FIFO1 is not full 1: The Rx FIFO1 is full
2	Reserved	Must be kept at reset value.
1:0	RFL1[1:0]	Rx FIFO1 length These bits are the length of the Rx FIFO1.

26.4.6. 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 disabled

		1: Sleep working interrupt enabled
16	WIE	Wakeup interrupt enable 0: Wakeup interrupt disabled 1: Wakeup interrupt enabled
15	ERRIE	Error interrupt enable 0: Error interrupt disabled 1: Error interrupt enabled
14:12	Reserved	Must be kept at reset value.
11	ERRNIE	Error number interrupt enable 0: Error number interrupt disabled 1: Error number interrupt enabled
10	BOIE	Bus-Off interrupt enable 0: Bus-Off interrupt disabled 1: Bus-Off interrupt enabled
9	PERRIE	Passive error interrupt enable 0: Passive error interrupt disabled 1: Passive error interrupt enabled
8	WERRIE	Warning error interrupt enable 0: Warning error interrupt disabled 1: Warning error interrupt enabled
7	Reserved	Must be kept at reset value.
6	RFOIE1	Rx FIFO1 overfull interrupt enable 0: Rx FIFO1 overfull interrupt disabled 1: Rx FIFO1 overfull interrupt enabled
5	RFFIE1	Rx FIFO1 full interrupt enable 0: Rx FIFO1 full interrupt disabled 1: Rx FIFO1 full interrupt enabled
4	RFNEIE1	Rx FIFO1 not empty interrupt enable 0: Rx FIFO1 not empty interrupt disabled 1: Rx FIFO1 not empty interrupt enabled
3	RFOIE0	Rx FIFO0 overfull interrupt enable 0: Rx FIFO0 overfull interrupt disabled 1: Rx FIFO0 overfull interrupt enabled
2	RFFIE0	Rx FIFO0 full interrupt enable 0: Rx FIFO0 full interrupt disabled 1: Rx FIFO0 full interrupt enabled
1	RFNEIE0	Rx FIFO0 not empty interrupt enable

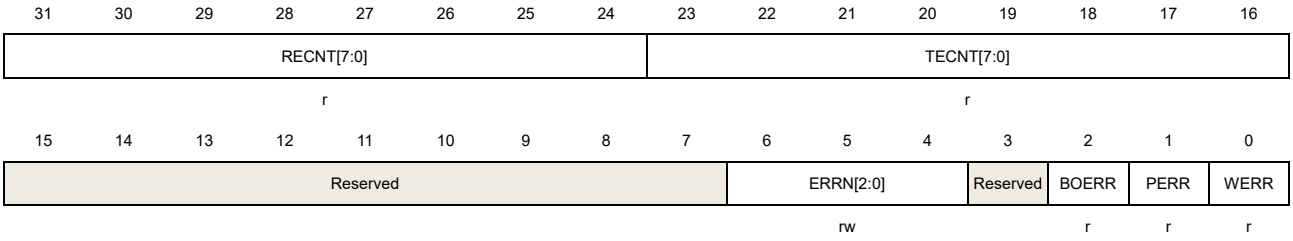
0: Rx FIFO0 not empty interrupt disabled
1: Rx FIFO0 not empty interrupt enabled

0 TMEIE Transmit mailbox empty interrupt enable
0: Transmit mailbox empty interrupt disabled
1: Transmit mailbox empty interrupt enabled

26.4.7. Error register (CAN_ERR)

Address offset: 0x18
Reset value: 0x0000 0000

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



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 hardware. When the bit transformation is successful, they are equal to 0. 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 hardware.
1	PERR	Passive error Whenever the TECNT or RECNT is greater than 127, the bit will be set by hardware.
0	WERR	Warning error

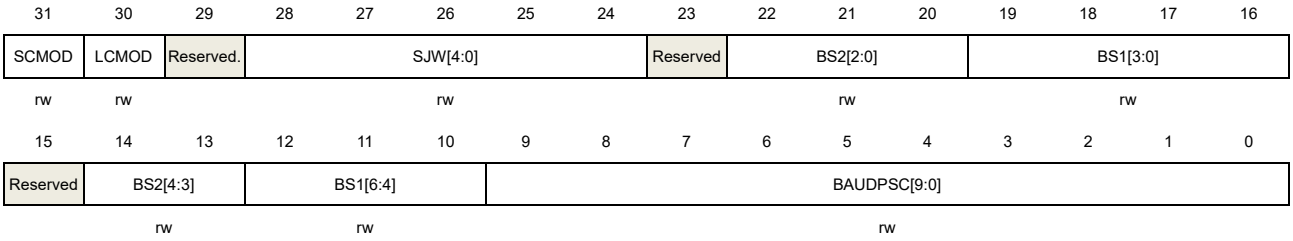
Whenever the TECNT or RECNT is greater than or equal to 96, the bit will be set by hardware.

26.4.8. Bit timing register (CAN_BT)

Address offset: 0x1C

Reset value: 0x0123 0000

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



Bits	Fields	Descriptions
31	SCMOD	Silent communication mode 0: Silent communication disabled 1: Silent communication enabled
30	LCMOD	Loopback communication mode 0: Loopback communication disabled 1: Loopback communication enabled
29	Reserved	Must be kept at reset value.
28:24	SJW[4:0]	Resynchronization jump width Resynchronization jump width time quantum = SJW[4:0]+1
23	Reserved	Must be kept at reset value.
22:20	BS2[2:0]	Bit segment 2 Bit segment 2 time quantum = BS2[4:0]+1
19:16	BS1[3:0]	Bit segment 1 Bit segment 1 time quantum = BS1[6:0]+1
15	Reserved	Must be kept at reset value.
14:13	BS2[4:3]	Bits 4:3 of BS2 See bits 22:20 of CAN_BT This bit can be configured only when FDEN=1 in CAN_FDCTL register. This bit is only available in GD32E508xx series.
12:10	BS1[6:4]	Bits 6:4 of BS1 See bits 19:16 of CAN_BT This bit can be configured only when FDEN=1 in CAN_FDCTL register.

This bit is only available in GD32E508xx series.

9:0	BAUDPSC[9:0]	Baud rate prescaler The CAN baud rate prescaler

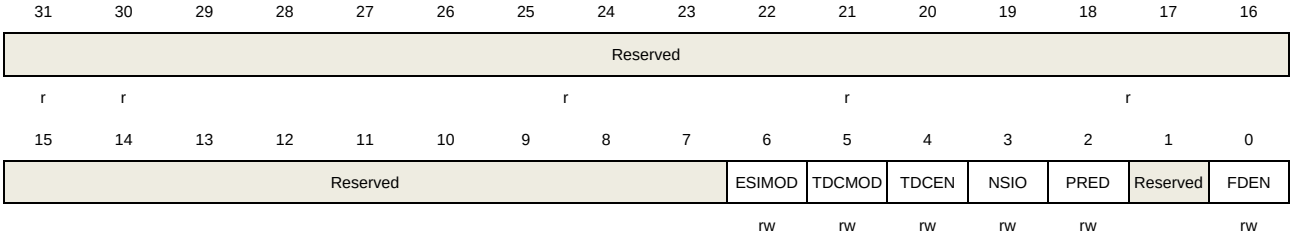
26.4.9. FD control register (CAN_FDCTL)

Address offset: 0x20

Reset value: 0x0000 0000

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

This register is only available in GD32E508xx series.



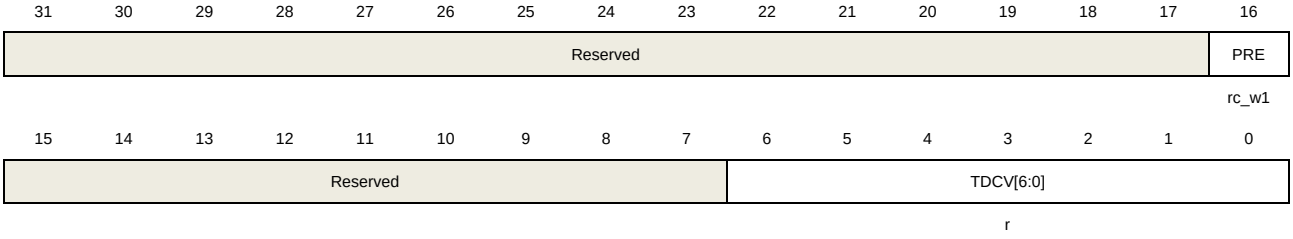
Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6	ESIMOD	Error state indicator mode 0: Always displays the node error state. Transmit the dominant bit by error active nodes and transmit the recessive bit by error passive nodes. 1: When the node is not in the error passive state: Displays the message buffer error state by configuring ESI bit in CAN_TMPx registers. When the node is in the error passive state: Displays the node error state.
5	TDCMOD	Transmitter delay compensation mode 0: Measurement and offset 1: Only offset
4	TDCEN	Transmitter delay compensation enable 0: Transmitter delay compensation is disabled 1: Transmitter delay compensation is enabled
3	NISO	ISO/Bosch 0: ISO 1: Bosch
2	PRED	Protocol exception event detection disable 0: Protocol exception event detection enabled (to idle) 1: Protocol exception event detection disabled (regarded as a form error)

1	Reserved	Must be kept at reset value.
0	FDEN	FD operation enable 0: CAN FD function disabled 1: CAN FD function enabled

26.4.10. FD status register (CAN_FDSTAT)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).
This register is only available in GD32E508xx series.

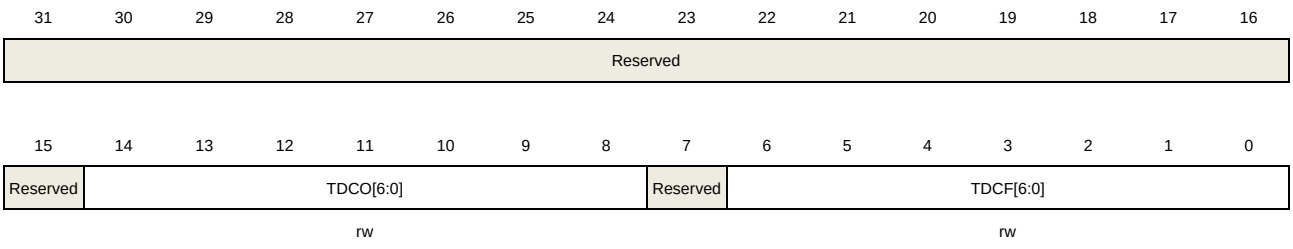


Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value.
16	PRE	Protocol exception event This bit is set by hardware when protocol exception event is detected, this bit is cleared by writting 1.
15:7	Reserved	Must be kept at reset value.
6:0	TDCV[6:0]	Transmitter delay compensation value These bits are set by hardware to display transmitter delay compensation value.

26.4.11. FD transmitter delay compensation register (CAN_FDTDC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).
This register is only available in GD32E508xx series.



Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14:8	TDCO[6:0]	Transmitter delay compensation offset These bits are set to the transmitter delay compensation offset value which defines the distance between the measured delay from CANTX to CANRX and the second sample point.
7	Reserved	Must be kept at reset value.
6:0	TDCF[6:0]	Transmitter delay compensation filter These bits define the minimum value for the SSP position. Dominant edges on CANRX that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCV.

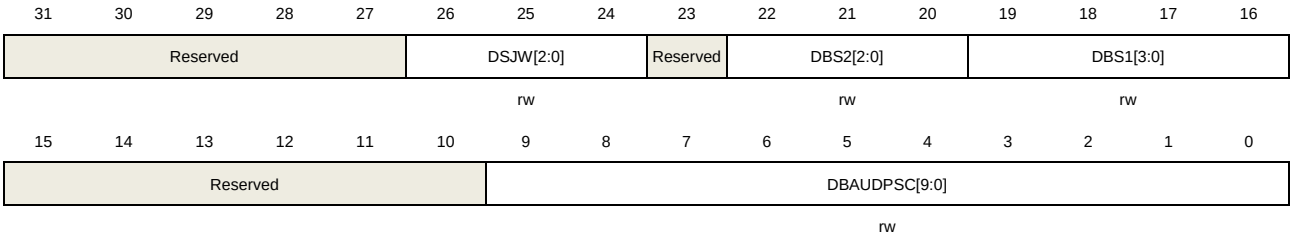
26.4.12. Date Bit timing register (CAN_DBT)

Address offset: 0x2C

Reset value: 0x0123 0000

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

This register is only available in GD32E508xx series.



Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26:24	DSJW[2:0]	Resynchronization jump width Resynchronization jump width time quantum = DSJW[2:0]+1
23	Reserved	Must be kept at reset value.
22:20	DBS2[2:0]	Bit segment 2 Bit segment 2 time quantum = DBS2[2:0]+1
19:16	DBS1[3:0]	Bit segment 1 Bit segment 1 time quantum = DBS1[3:0]+1
15:10	Reserved	Must be kept at reset value.
9:0	DBAUDPSC[9:0]	Baud rate prescaler The CAN baud rate prescaler

26.4.13. Transmit mailbox identifier register (CAN_TMIx) (x = 0…2)

Address offset: 0x180 + 0x10 * x

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 software when one frame will be transmitted and reset by hardware when the transmit mailbox is empty. 0: Transmit disabled 1: Transmit enabled

26.4.14. Transmit mailbox property register (CAN_TMPx) (x = 0…2)

Address offset: 0x184 + 0x10 * x

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]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							TSEN	FDF	Reserved	BRS	ESI	DLENC[3:0]			
							rw	rw		rw	rw	rw			

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 disabled 1: Time stamp enabled. The TS[15:0] will be transmitted in the DB6 and DB7 in DL. This bit is available when the TTC bit in CAN_CTL is set.
7	FDF	CAN FD frame flag 0: Classical frames 1: FD frames This bit is only available in GD32E508xx series.
6	Reserved	Must be kept at reset value.
5	BRS	Bit rate of data switch 0: Bit rate not switch 1: The bit rate shall be switched from the nominal bit rate of the arbitration phase to the preconfigured bit rate of data of the data phase This bit is only available in GD32E508xx series.
4	ESI	Error status indicator This bit is valid when ESIMOD bit is 1 in CAN_FDCTL register 0: Transmit the dominant bit in ESI phase 1: Transmit the recessive bit in ESI phase This bit is only available in GD32E508xx series.
3:0	DLENC[3:0]	Data length code DLENC[3:0] is the number of bytes in a frame

26.4.15. Transmit mailbox data0 register (CAN_TMDATA0x) (x = 0…2)

Address offset: 0x188 + 0x10 * x

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]							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB1[7:0]								DB0[7:0]							
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

26.4.16. Transmit mailbox data1 register (CAN_TMDATA1x) (x = 0...2)

Address offset: 0x18C + 0x10 * x
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

26.4.17. Receive FIFO mailbox identifier register (CAN_RFIFOMIx) (x = 0,1)

Address offset: 0x1B0 + 0x10 * x
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.

26.4.18. Receive FIFO mailbox property register (CAN_RFIFOMPx) (x = 0,1)

Address offset: 0x1B4 + 0x10 * x

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]								FDF	Reserved	BRS	ESI	DLENC[3:0]			
r								rw		rw	rw	r			

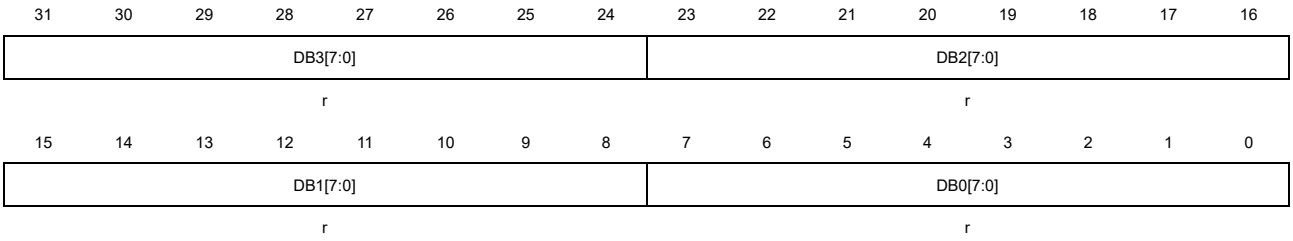
Bits	Fields	Descriptions
31:16	TS[15:0]	Time stamp The time stamp of frame in transmit mailbox.
15:8	FI[7:0]	Filtering index The index of the filter which the frame passes.
7	FDF	CAN FD frame flag 0: Classical frames 1: FD frames This bit is only available in GD32E508xx series.

6	Reserved	Must be kept at reset value.
5	BRS	Bit rate of data switch 0: Bit rate not switch 1: The bit rate shall be switched from the nominal bit rate of the arbitration phase to the preconfigured bit rate of data of the data phase This bit is only available in GD32E508xx series.
4	ESI	Error status indicator This bit is valid when ESIMOD bit is 1 in CAN_FDCTL register 0: Transmit the dominant bit in ESI phase 1: Transmit the recessive bit in ESI phase This bit is only available in GD32E508xx series.
3:0	DLENC[3:0]	Data length code DLENC[3:0] is the number of bytes in a frame

26.4.19. Receive FIFO mailbox data0 register (CAN_RFIFOMDATA0x) (x = 0,1)

Address offset: 0x1B8 + 0x10 * x
Reset value: 0xXXXX XXXX

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

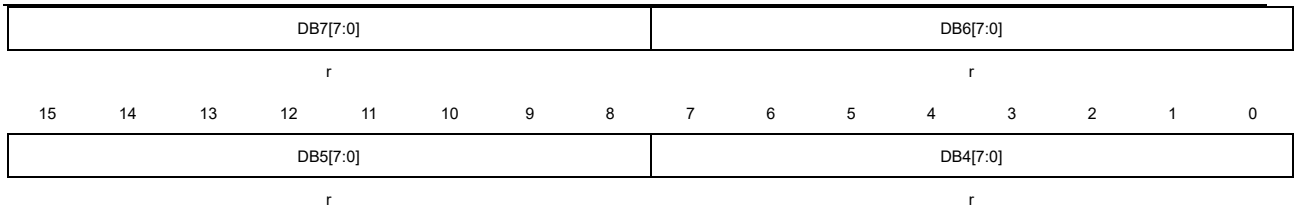


Bits	Fields	Descriptions
31:24	DB3[7:0]	Data byte 3
23:16	DB2[7:0]	Data byte 2
15:8	DB1[7:0]	Data byte 1
7:0	DB0[7:0]	Data byte 0

26.4.20. Receive FIFO mailbox data1 register (CAN_RFIFOMDATA1x) (x = 0,1)

Address offset: 0x1BC + 0x10 * x
Reset value: 0xXXXX XXXX

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



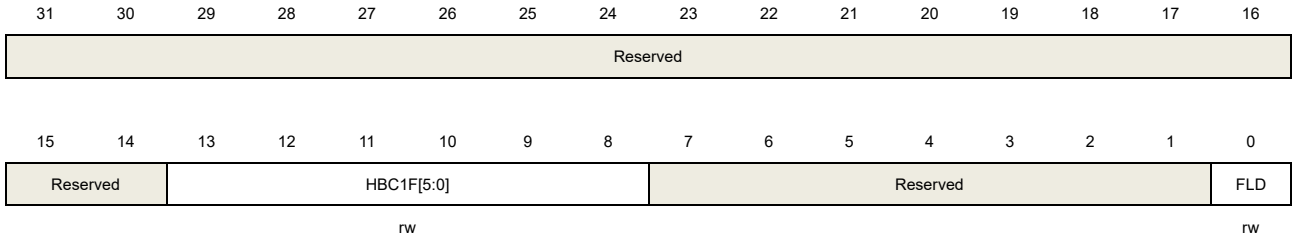
Bits	Fields	Descriptions
31:24	DB7[7:0]	Data byte 7
23:16	DB6[7:0]	Data byte 6
15:8	DB5[7:0]	Data byte 5
7:0	DB4[7:0]	Data byte 4

26.4.21. Filter control register (CAN_FCTL)

Address offset: 0x200

Reset value: 0x2A1C 0E01

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



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 is used for CAN0. Bank HBC1F ~ Bank27 is used for CAN1. When set 0, no bank used for CAN0. When set 28, no bank used for CAN1. These bits are only used in CAN0 and CAN1.
7:1	Reserved	Must be kept at reset value.
0	FLD	Filter lock disable 0: Filter lock enabled 1: Filter lock disabled

26.4.22. 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.

The filter mode configuration register inCAN0 and CAN1:

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

The filter mode configuration register inCAN2:

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		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

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:0	FMODx	Filter mode 0: Filter x with mask mode 1: Filter x with list mode

26.4.23. 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.

The filter scale configuration register in CAN0 and CAN1:

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

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

The filter scale configuration register in CAN2:

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		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

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:0	FSx	Filter scale 0: Filter x with 16-bit scale 1: Filter x with 32-bit scale

26.4.24. 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.
The filter associated FIFO register in CAN0 and CAN1:

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

The filter associated FIFO register in CAN2:

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		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

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:0	FAFx	Filter associated FIFO 0: Filter x associated with FIFO0 1: Filter x associated with FIFO1

26.4.25. Filter working register (CAN_FW)

Address offset: 0x21C

Reset value: 0x0000 0000

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

The filter working register in CAN0 and CAN1:

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 disabled 1: Filter x working enabled

The filter working register in CAN2.

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		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

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:0	FWx	Filter working 0: Filter x working disabled 1: Filter x working enabled

26.4.26. Filter x data y register (CAN_FxDATAy) (x = 0…27, y = 0,1)

Address offset: 0x240 + 8 * x + 4 * y, (In CAN2: x = 0..13, 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

27. Ethernet (ENET)

27.1. Overview

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 datas 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 1588-2008.

27.2. 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.
- Software configurable for inter-frame gap.
- 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™ frames.
- Support checking checksum (IPv4 header, TCP, UDP or ICMP encapsulated in IPv4 or IPv6 data format).
- Support Ethernet frame time stamping for both transmit and receive operation, which describes in IEEE 1588-2008, and 64 bits time stamps are given in each frame’s status.
- Two independent FIFO for transmitting and receiving.

- Support special condition frame discards handling, e.g. late collision, excessive collisions, excessive deferral or underrun.
- In the process of frame transmission, support computation and insertion of hardware checksum under store-and-forward mode.

DMA Feature

- Two types of descriptor addressing: Ring and Chain.
- Descriptor of transmit and receive both can transfer data up to 8192 bytes.
- Software configurable normal and abnormal interrupt for many status conditions.
- Support round-robin or fixed priority to arbitrate the request of transmit and receive controller.

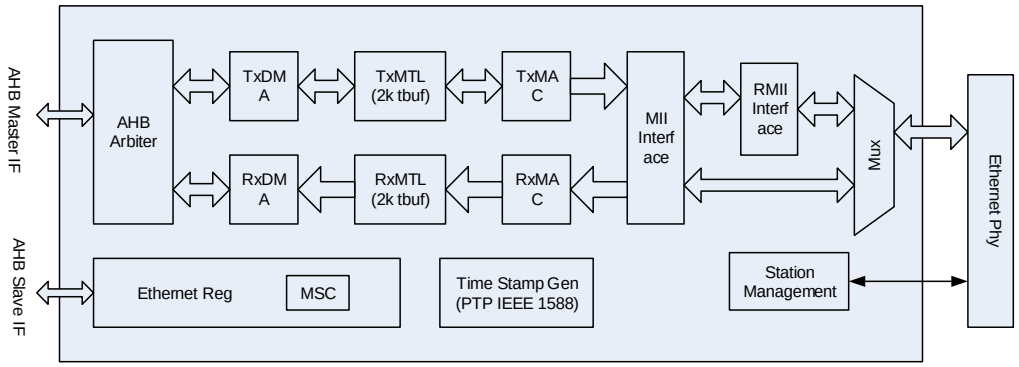
PTP Feature

- Support IEEE 1588 time synchronization function.
- Support two correction methods: Coarse or Fine.
- Support output pulse in seconds.
- Preset expected time reaching trigger and interrupt

27.2.1. Block diagram

The Ethernet module is composed of a MAC module, MII / RMII module and a DMA module by descriptor control. When using Ethernet, the user should ensure that the configured AHB clock frequency is no less than 25MHz.

Figure 27-1. ENET module block diagram



The MAC module is connected to the external PHY by MII or RMII through bit ENET_PHY_SEL in 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.

27.2.2. MAC 802.3 Ethernet packet description

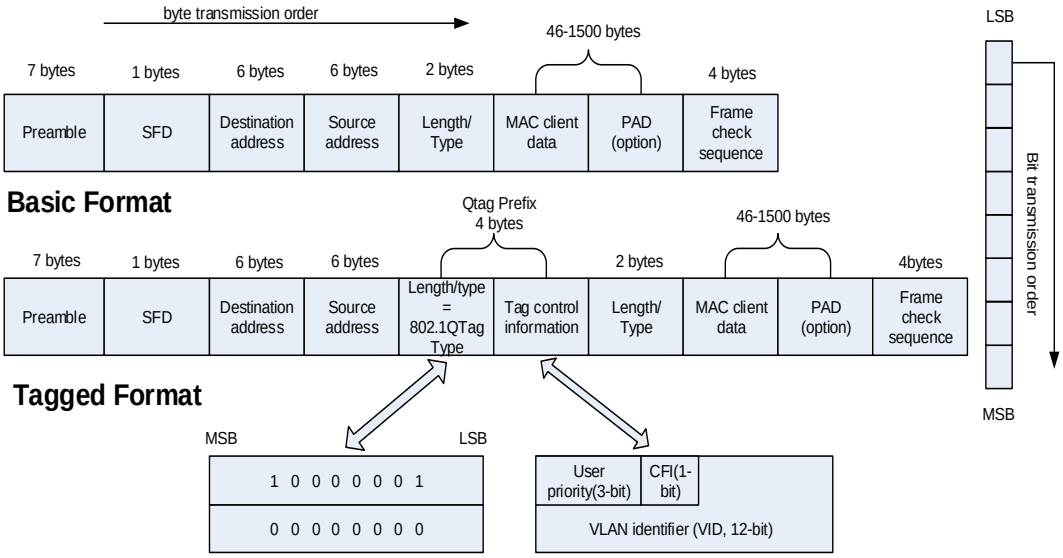
Data communication of MAC can use two frame formats:

- Basic frame format.
- Tagged frame format.

Figure 27-2. MAC / Tagged MAC frame **format**

describes the structure of the frame (Basic and Tagged) that includes the following fields:

Figure 27-2. 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:

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

27.2.3. Ethernet signal description

Table 27-1. Ethernet signals (MII default) / *Table 27-2. Ethernet signals (MII remap)* / *Table 27-3. Ethernet signals (RMII default)* and *Table 27-4. Ethernet signals (RMII remap)* shows the MAC module that pin is used default and remapping functions and specific configuration in MII / RMII mode.

Table 27-1. Ethernet signals (MII default)

Signals	Pin	Pin mode
MDC	PC1	AF output push-pull
MII_TXD2	PC2	AF output push-pull
MII_TX_CLK	PC3	Floating input (reset state)
MII_CRS	PA0	Floating input (reset state)
MII_RX_CLK	PA1	Floating input (reset state)
MDIO	PA2	AF output push-pull
MII_COL	PA3	Floating input (reset state)
MII_RX_DV	PA7	Floating input (reset state)
MII_RXD0	PC4	Floating input (reset state)
MII_RXD1	PC5	Floating input (reset state)
MII_RXD2	PB0	Floating input (reset state)
MII_RXD3	PB1	Floating input (reset state)
PPS_OUT	PB5	AF output push-pull
MII_TXD3	PB8	AF output push-pull
MII_RX_ER	PB10	Floating input (reset state)
MII_TX_EN	PB11	AF output push-pull
MII_TXD0	PB12	AF output push-pull
MII_TXD1	PB13	AF output push-pull

Table 27-2. Ethernet signals (MII remap)

Signals	Pin	Pin mode
MDC	PC1	AF output push-pull
MII_TXD2	PC2	AF output push-pull
MII_TX_CLK	PC3	Floating input (reset state)
MII_CRS	PA0	Floating input (reset state)
MII_RX_CLK	PA1	Floating input (reset state)
MDIO	PA2	AF output push-pull
MII_COL	PA3	Floating input (reset state)
MII_RX_DV	PD8	Floating input (reset state)
MII_RXD0	PD9	Floating input (reset state)

Signals	Pin	Pin mode
MII_RXD1	PD10	Floating input (reset state)
MII_RXD2	PD11	Floating input (reset state)
MII_RXD3	PD12	Floating input (reset state)
PPS_OUT	PB5	AF output push-pull
MII_TXD3	PB8	AF output push-pull
MII_RX_ER	PB10	Floating input (reset state)
MII_TX_EN	PB11	AF output push-pull
MII_TXD0	PB12	AF output push-pull
MII_TXD1	PB13	AF output push-pull

Table 27-3. Ethernet signals (RMII default)

Signals	Pin	Pin mode
MDC	PC1	AF output push-pull
REF_CLK	PA1	Floating input (reset state)
MDIO	PA2	AF output push-pull
CRS_DV	PA7	Floating input (reset state)
RMII_RXD0	PC4	Floating input (reset state)
RMII_RXD1	PC5	Floating input (reset state)
PPS_OUT	PB5	AF output push-pull
RMII_TX_EN	PB11	AF output push-pull
RMII_TXD0	PB12	AF output push-pull
RMII_TXD1	PB13	AF output push-pull

Table 27-4. Ethernet signals (RMII remap)

Signals	Pin	Pin mode
MDC	PC1	AF output push-pull
REF_CLK	PA1	Floating input (reset state)
MDIO	PA2	AF output push-pull
CRS_DV	PD8	Floating input (reset state)
RMII_RXD0	PD9	Floating input (reset state)
RMII_RXD1	PD10	Floating input (reset state)
PPS_OUT	PB5	AF output push-pull
RMII_TX_EN	PB11	AF output push-pull
RMII_TXD0	PB12	AF output push-pull
RMII_TXD1	PB13	AF output push-pull

27.3. Function overview

27.3.1. 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

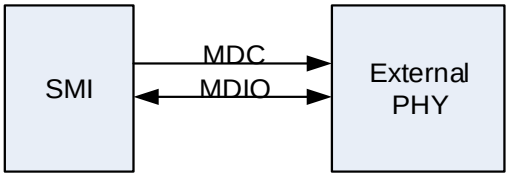
SMI is designed to access and configure PHY's configuration.

Station management interface (SMI) is performed through two wires to communicate with the external PHY: one clock line (MDC) and one data line (MDIO) . The maximum number of PHYs supported by this interface is 32. But at the same time only one register of a PHY can be accessed.

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 27-3. Station management interface signals



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.

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.

clock configuration

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. ***Table 27-5. Clock range*** lists the frequency factor corresponding AHB clock selection.

Table 27-5. Clock range

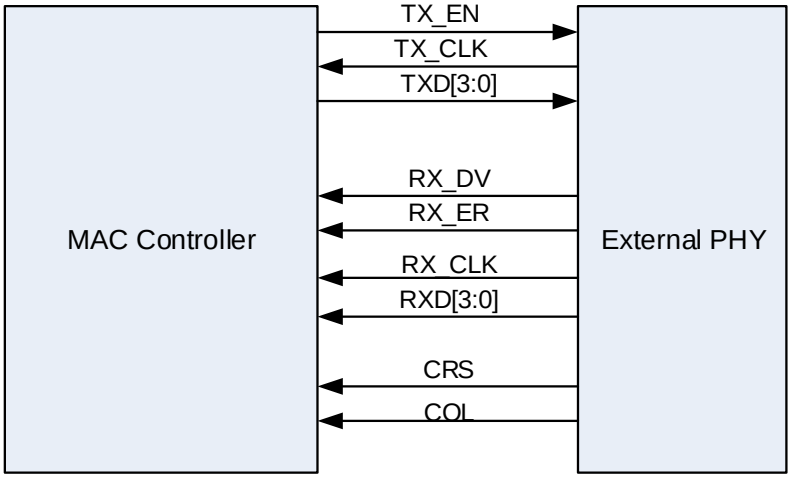
AHB clock	MDC clock	Bits CLR[2:0] in ENET_MAC_PHY_CTL
150~180MHz	AHB clock / 102	0x4
35~60MHz	AHB clock / 26	0x3
20~35MHz	AHB clock / 16	0x2
100~150 MHz	AHB clock / 62	0x1
60~100MHz	AHB clock / 42	0x0

MII / RMII selection

Before enable the Ethernet controller clocks or when the Ethernet controller is under the reset state, the application can select the MII or RMII mode by configuring ENET_PHY_SEL in the AFIO_PCF0 register The MII mode is set by default.

MII: Media independent interface

Figure 27-4. 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. This signal must be active when the first bit of the data preamble occurs. And it needs to remain active before the all bits transmission is completed.

- **MII_TXD[3:0]**: Transmit data line, each 4 bit data transfer, data is valid when the MII_TX_EN signal is effective. The PHY would ignore the transmitted data when the MII_TX_EN signal is non-effective.

- **MII_CRS**: Carrier sense signal, only working in Half-duplex mode and controlled by the PHY. This signal does not need to be synchronized with the MII_TX_CLK and MII_RX_CLK. When it is active, means that the transmit or receive medium is not in idle state. MII_CRS signal remains active until the transmit and receive medium are both in idle state.

- **MII_COL**: Collision detection signal, only working in Half-duplex mode, controlled by the PHY. This signal does not need to be synchronized with the MII_TX_CLK and MII_RX_CLK. It is active when a collision on the medium is detected and it will remain active while the collision condition continues.

- **MII_RXD[3:0]**: Receive data line, each 4 bit data transfer; data are valid when the MII_RX_DV signal is effective. Depending on the state of MII_RX_DV and MII_RX_ER, the MII_RXD[3:0] value can be used to convey some specific information (see ***Table 27-6. Rx interface signal encoding***).

- **MII_RX_DV**: Receive data valid signal, controlled by the PHY. This signal must be active when the first 4-bits of the frame data occurs. And it needs to remain active before the all bits transmission is completed. It must be inactive prior to the first clock cycle that follows the final 4-bit. MII_RX_DV signals should be effective before the SFD field appearing to ensure that receive the correct frame.

- **MII_RX_ER**: Receive error signal. In order to indicate that MAC detected an error in the receiving process, the MII_RX_ER signal must remain effective for one or more clock cycles (MII_RX_CLK). 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 27-6. Rx interface signal encoding***).

Table 27-6. Rx interface signal encoding

Signal	Normal inter-frame		Normal reception frame data	False carrier indication	Data reception with errors
MII_RX_ER	0	1	0	1	1
MII_RX_DV	0	0	1	0	1
MII_RXD[3:0]	0000 to 1111	0000	0000 to 1111	1110	0000 to 1111

MII clock sources

The user needs to provide an external 25MHz clock to the external PHY to generate both TX_CLK and RX_CLK 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.

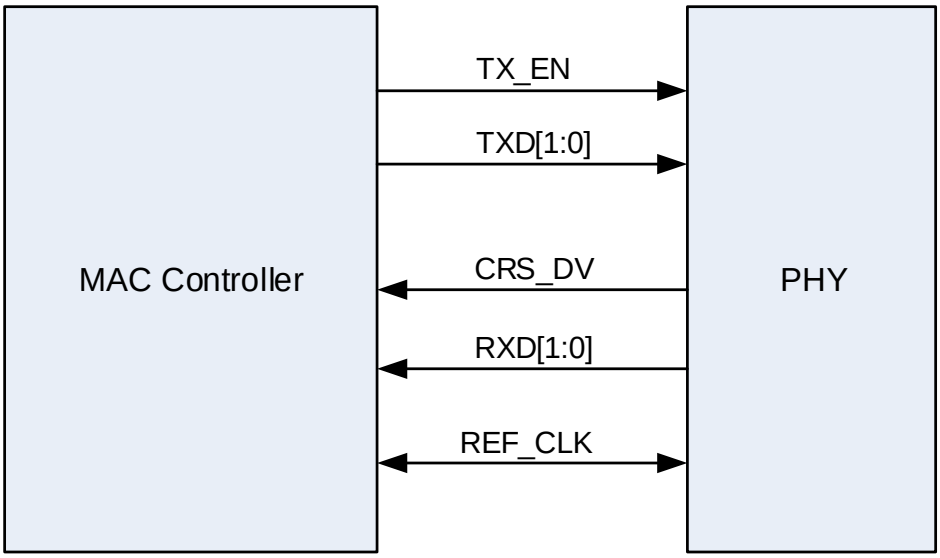
RMII: Reduced media independent interface

The reduced media-independent interface (RMII) specification reduces the pin count during Ethernet communication. The MII specification defines 16 pins for data and control, according to the IEEE 802.3 standard. The RMII specification is dedicated to reduce the pin count to 7.

RMII 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 27-5. Reduced media-independent interface signals



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.

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, 1101 corresponding to MII_T/RXD[3] to MII_T/RXD[0]).
Transmission order for RMII use 4 cycles: 01 -> 11 -> 01 -> 10 (left to right order: high to low, 01 corresponding to RMII_T/RXD[1] to RMII_T/RXD[0]).

27.3.2. MAC function overview

The MAC module can work in two modes (Half-duplex mode and Full-duplex mode). The 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. The 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.

MAC module can achieve the follows functions: 1) The data packaging (transmission and reception), that includes detecting / decoding frame and delimitating frame boundary; handling source address and destination address; detecting error conditions. 2) The Medium access management in Half-duplex mode, that includes allocating medium in order to prevent conflicts; deal with conflicts.

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 sees 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, the data in FIFO is ready to be popped to MAC once the number of bytes in the FIFO exceeds or equals the configured threshold level or when the end-of-frame flag in descriptor is written. User can configure the threshold level through the TTHC[2:0] in ENET_DMA_CTL
- In Store-and-forward mode, the data is ready to be popped to the MAC core only after complete frame is stored in the FIFO. But there is another condition where the frame is not completely written into the FIFO, and FIFO will also take out data. This is when the transmitted Ethernet frame is bigger than FIFO size, the frame is popped towards the MAC before the transmit FIFO becomes 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 FIFO flush operation

Application can clear TxFIFO and reset the FIFO data pointer by setting FTF bit 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 (FRMF and UFE bits in Transmit Descriptor0). 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.

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:

- The carrier sense signal active in the first 2 / 3 gap period. In this case, the counter will reload and restart.
- 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.
- 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.

Address receive module

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.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.

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 Receive Descriptor0 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.

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.

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.

Hash or perfect address filter

By setting the HPFLT bit in the ENET_MAC_FRMF register and setting the corresponding HUF (for unicast frame) or HMF (for multicast frame) bit in the ENET_MAC_FRMF register, 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.

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 ***Table 27-7. Destination address filtering table*** and ***Table 27-8. Source address filtering table*** summarize the destination address and source address filters working condition at different configuration.

Table 27-7. 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 27-8. 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, the automatically strip FCS field function is configured by the TFCD bit regardless of APCD.

If the watchdog timer is enabled (WDD bit in ENET_MAC_CFG is reset), a frame has more than 2048 bytes 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, 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.

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.

Flow control module

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.

The MAC manages reception frames through follow method for 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.

Checksum offload engine

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.

The follows describes the operation of 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.

Refer to 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 (In Transmit Descriptor0). The following shows the conditions under which the IPHE bit can be set:

- 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).

- For IPv6 frame type:

A) . Type field is 0x86dd but version field in IP header is not 0x6.
B) . Before the IPv6 standard header or extension header has been completely received the frame is end. The length of IPv6 standard header is 40 bytes, and the extension header contains corresponding header length field.

- 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:

- Incomplete IPv4 or IPv6 frames.
- IP frames with security features (e.g. authentication header, security payload).
- IP frames without TCP / UDP / ICMPv4 / ICMPv6 payload.
- IPv6 frames with routing headers.

The checksum offload module calculates the payload (TCP, UDP, or ICMP) and inserts the result into its corresponding field in the header. It has two modes when working, as follows:

1. The checksum calculation does not include TCP, UDP, or ICMPv6 pseudo-headers and assumes that the checksum field of the input frame already has the value. The checksum calculation includes checksum field, and the value of the original checksum field is replaced after the calculation is completed.

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 of Transmit Descriptor0. 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.The follows describes the operation of 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 IPHERR bit in Receive Descriptor0) 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 (the PCERR bit in Receive Descriptor0) 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.

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.

27.3.3. 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 two kinds of data structures. Which are descriptor table (ring or chain type) and data buffer, and 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 (Transmit Descriptor0-3) when DFM=0 and 8 descriptor word (Transmit Descriptor0-7) when DFM=1 (Enhanced descriptor mode). Likewise, reception descriptors constituted by 4 descriptor word (Receive Descriptor0-3) when DFM=0 and 8 descriptor word (Receive Descriptor0-7) when DFM=1. 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 (configure RCHM bit in the Receive Descriptor1 and TCHM bit in the Transmit Descriptor0), at this time Receive Descriptor2 and Transmit Descriptor2 are stored the data buffer address, Receive Descriptor3 and Transmit Descriptor3 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 RCHM bit in the Receive

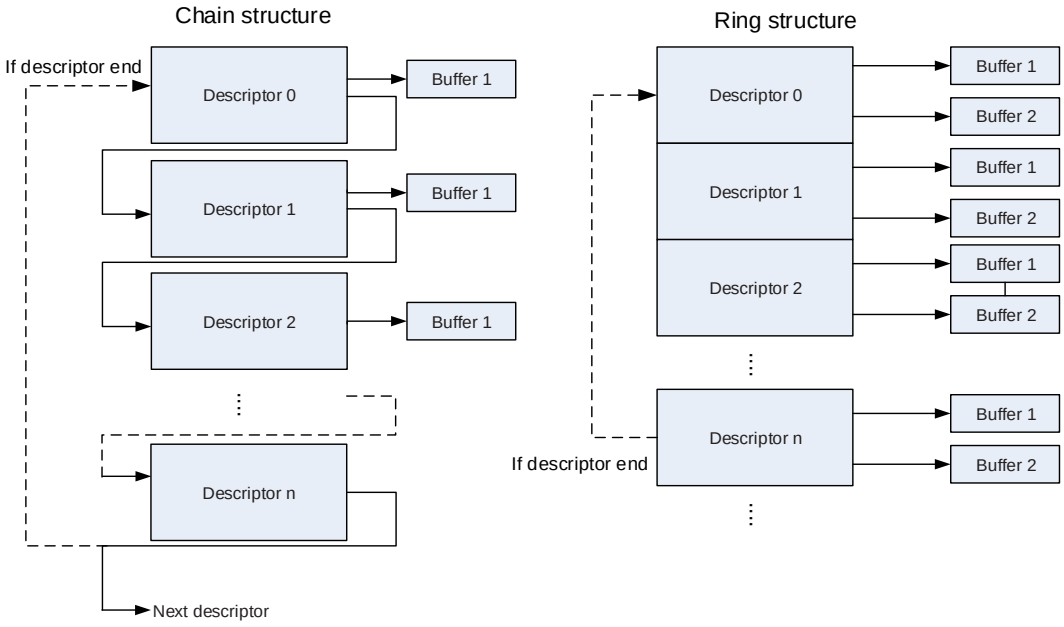
Descriptor1 and TCHM bit in the Transmit Descriptor0, at this time Receive Descriptor2/Transmit Descriptor2 and Receive Descriptor3/Transmit Descriptor3 should be all stored the data buffer address, this connection method of descriptor table is called ring structure. When current descriptor's buffer 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:

DFM=0: Next descriptor address = Current descriptor address + 16 + DPSL * 4

DFM=1: Next descriptor address = Current descriptor address + 32 + DPSL * 4

If current descriptor is the last one in descriptor table, application needs to set the TERM bit in Transmit Descritor0 or RERM bit in Receive Descritor1 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 Receive Descriptor3 and Transmit Descriptor3 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 27-6. 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 Transmit Descriptor1. 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 Transmit Descriptor0 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 Transmit Descriptor0 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 Transmit Descriptor0 is set, it knows 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 Receive Descriptor1. When the receive buffer address is not word align, the effective length of the buffer is less than the value configured by application in Receive Descriptor1. 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 Receive Descriptor0. 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.

DMA error status

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.

Transmit process of DMA

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.

Transmission management of DMA

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 (Transmit Descriptor0-3) with DAV bit of Transmit Descriptor0 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 Transmit Descriptor0 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 Transmit Descriptor0 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 Transmit Descriptor0 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 a 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 Transmit Descriptor0 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.

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

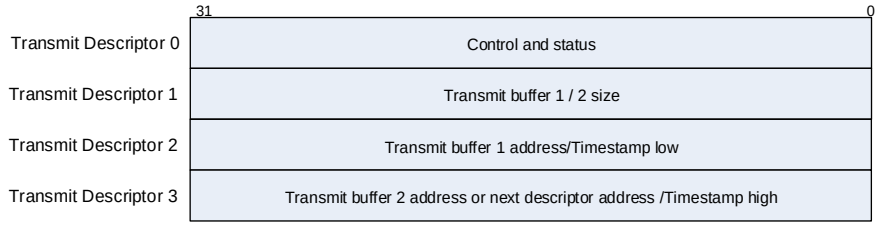
When TTSEN bit is set, the timestamp function is enabled. The TxDMA controller writes transmit timestamp status TTMSS and timestamp back to descriptor after the frame transmission complete. The word address in descriptor for writing timestamp is depends on DFM bit in ENET_DMA_BCTL register. If the descriptor format is normal mode (DFM=0), Transmit Descriptor2 and Transmit Descriptor3 are used for timestamp recording and the old values in Transmit Descriptor2 and Transmit Descriptor3 are overwritten. If the descriptor format is enhanced mode (DFM=1), Transmit Descriptor6 and Transmit Descriptor7 are used for timestamp recording and the value in Transmit Descriptor2 and Transmit Descriptor3 are kept.

TxDMA descriptors in normal mode

The normal mode descriptor structure consists of four 32-bit words: Transmit Descriptor0 ~ Transmit Descriptor3**.** The descriptions of Transmit Descriptor0 ~ Transmit Descriptor3 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 27-7. Transmit descriptor in normal mode



■ Transmit Descriptor0

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 Only when the LSG bit is set, this bit is valid. 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 shows whether the transmit 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 shows whether 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 Only when the FSG bit is set, this bit is valid. 0: Allow MAC to insert CRC at the end of transmitted frame automatically 1: Not Allow MAC to insert CRC at the end of transmitted frame
26	DPAD	Disable adding pad bit Only when the FSG bit is set, this bit is valid. 0: The DMA adds padding byte and CRC to transmitted frame automatically. Only the padding actually acts, the CRC is also appended. And ignore the value of DCRC bit. 1: The MAC does not add padding to a frame automatically
25	TTSEN	Transmit timestamp function enable bit. Only when the FSG bit is set, this bit is valid.

		0: Disable transmit timestamp function 1: Enable IEEE 1588 hardware time stamping for the transmit frame, when TMSEN bit in the ENET_PTP_TSCTL register is set.
24	Reserved	Must be kept at reset value.
23:22	CM[1:0]	Checksum mode bits 0x0: Disable 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, hardware does not calculate checksum of pseudo-header. 0x3: Enable IP Header checksum and payload checksum calculation and insertion, hardware calculates checksum of pseudo-header.
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 TCHM bit is set, TB2S[12:0] is ignored. 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 Only when the LSG bit is set, this bit is valid. 0: Timestamp was not captured 1:A timestamp was captured for the described transmit frame and push into Transmit Descriptor2 (or Transmit Descriptor6 if DFM=1) and Transmit Descriptor 3 (or Transmit Descriptor7if DFM=1)
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: IPHE: IP header error JT: Jabber timeout FRMF: Frame flush IPPE: IP payload error LCA: Loss of carrier NCA: No carrier LCO: Late collision ECO: Excessive collision EXD: Excessive deferral UFE: Underflow error
14	JT	Jabber timeout bit Only set when the JBD bit is reset. 0: No jabber timeout occurred 1: Jabber timeout of MAC transmitter has occurred
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. Only in Half-duplex mode this bit is valid. 0: No loss of carrier occurred 1: When the frame is transmitting, loss of carrier occurred
10	NCA	No carrier bit 0: PHY carrier sense signal is active 1: When the frame is transmitting, the carrier sense signal from the PHY was not active
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 Only when ECO bit is cleared, this bit is valid. Before the frame was transmitted, this 4-bit counter counts the number of collisions that has occurred.
2	EXD	Excessive deferral bit Only when the DFC bit in the ENET_MAC_CFG register is set, this bit is valid. 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 shows 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 shows whether the transmitting frame is deferred because of interface signal CRS is active before MAC transmit frame. Only in Half-duplex mode this bit is valid. 0: No transmission deferred 1:The MAC is deferred before transmission

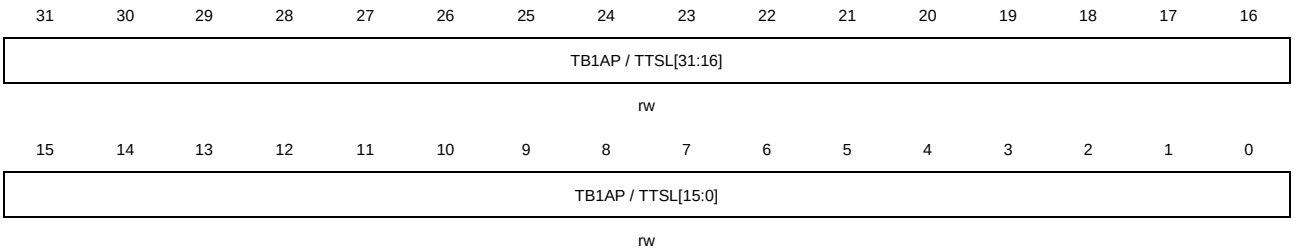
- Transmit descriptor1

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												

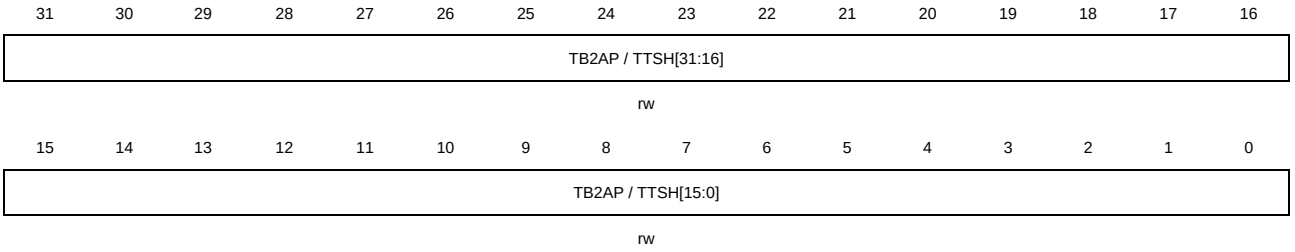
Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value.
28:16	TB2S[12:0]	Transmit buffer 2 size bits The second data buffer byte size.
15:13	Reserved	Must be kept at reset value.
12:0	TB1S[12:0]	Transmit buffer 1 size bits The first data buffer byte size.

■ Transmit descriptor2



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 if DFM=0. But if DFM=1, these bits will not change and keep the value of buffer address. 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.

■ Transmit descriptor word 3



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 if DFM=0

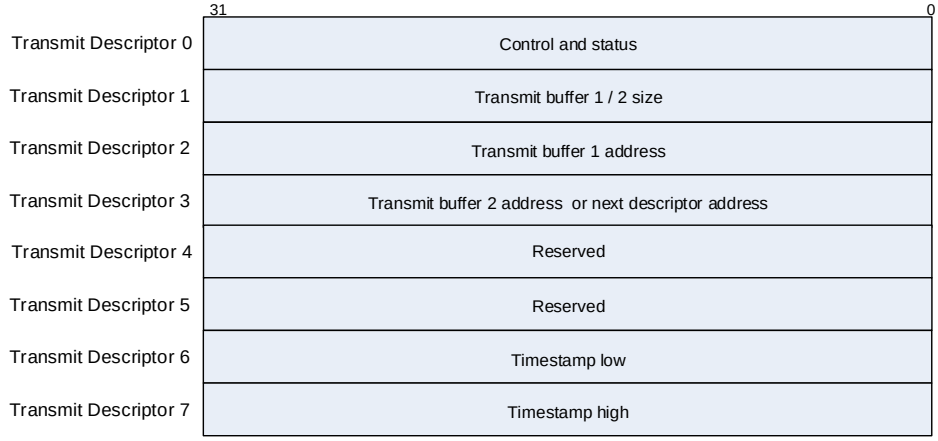
and TTSEN =1. But if DFM=1 or TTSEN =0, these bits will not change and keep the old value. When these bits stand for buffer 2 address (TCHM=0), the alignment is no limitation. When these bits stand for the next descriptor address (TCHM=1), these bits must be word-alignment. When these bits stand for timestamp high 32-bit value, the TTSEN and LSG bit of current descriptor must be set.

TxDMA descriptors in enhanced mode

The enhanced mode descriptor structure consists of eight 32-bit words: Transmit Descriptor0 ~ Transmit Descriptor7. The descriptions of Transmit Descriptor0 ~ Transmit Descriptor3. are the same with normal mode descriptor; Transmit Descriptor4 ~ Transmit Descriptor7. 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 DMA controller (except INTC). But the status and timestamp (if enabled) are written back to the last descriptor.

Figure 27-8. Transmit descriptor in enhanced mode



- Transmit descriptor4
  All bits reserved.
- Transmit descriptor5
  All bits reserved.
- Transmit descriptor6

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

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

Bits	Fields	Descriptions

31:0	TTSL[31:0]	Transmit frame timestamp low 32-bit value bits When TTSEN =1 and LSG=1, there bits are updated by TxDMA for recording timestamp low 32-bit value of the current transmitting frame.

- Transmit descriptor 7

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

Bits	Fields	Descriptions
31:0	TTSH[31:0]	Transmit frame timestamp high 32-bit value bits When TTSEN =1 and LSG=1, these bits are updated by TxDMA for recording timestamp high 32-bit value of the current transmitting frame.

Reception process of DMA

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:

- The received frame bytes is less than 64.
- Collision occurred during frame receiving.
- 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 Receive Descriptor0 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 Receive Descriptor0 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 Receive Descriptor0 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 (immediately when DINTC=0 or delayed when DINTC=1). 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.

Reception management of DMA

The receiving process of the RxDMA controller is described detailed as below:

1. Applications initialize the receive descriptors with the DAV bit in the Receive Descriptor0 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:
   - The next descriptor's DAV bit is reset. The RxDMA controller sets descriptor error bit DERR in Receive Descriptor0 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.
   - 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 Receive Descriptor2 and Receive Descriptor3 if DFM=0 or Receive Descriptor6 and Receive Descriptor7 if DFM=1. At the same time (writing timestamp value) the RxDMA controller also writes the received frame's status word to the Receive Descriptor0 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.

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 Receive Descriptor0 is set, the RxDMA controller exits suspend state and returns to running state for frame reception. But if the DAV bit in Receive Descriptor0 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

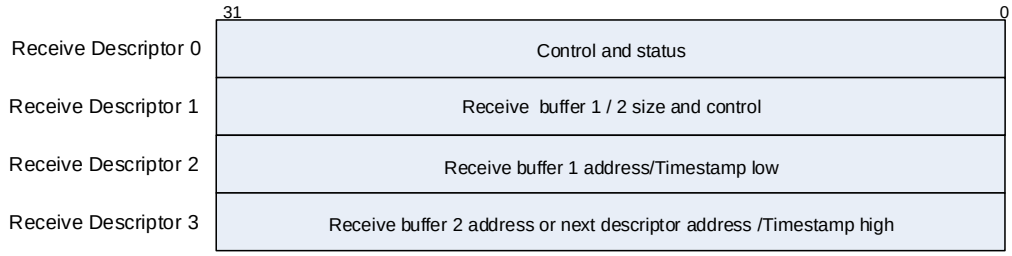
If the IEEE 1588 function enabled, the MAC writes the timestamp value to Receive Descriptor2 and Receive Descriptor3 (DFM=0) or Receive Descriptor6 and Receive Descriptor7 (DFM=1) after a frame with timestamp reception complete and before the RxDMA controller clears the DAV bit.

RxDMA descriptors in normal mode

In normal descriptor mode, the descriptor structure consists of four 32-bit words: Receive Descriptor0 ~ Receive Descriptor3 The detailed description of Receive Descriptor0 ~ Receive

Descriptor3 are given below.

Figure 27-9. Receive descriptor in normal mode



■ Receive descriptor 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 / TSV	LCO	FRMT	RWDT	RERR	DBERR	CERR	PCERR / EXSV
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 shows 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 show the byte length of the received frame that was transferred to the buffer (including CRC when received frame is not a type frame. If received frame is a type frame, including CRC or not is controlled by TFCD bit in ENET_MAC_CFG). Only when the bit LDES=1 and DERR=0, these bits are valid. If LDES=0 and ERRS=0, these bits indicate the accumulated number of bytes that have been transferred for the current frame. **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.
15	ERRS	Error summary bit Only when the LDES bit is set, this bit is valid.

		This bit is logical ORed by the following bits when DFM is equal to 0: DERR: Descriptor error OERR: Overflow error LCO: Late collision RWDT: Watchdog timeout RERR: Receive error CERR: CRC error IPHERR = 0, FRMT = 1 and PCERR = 1: payload checksum error IPHERR = 1, FRMT = 1 and PCERR = 0: header checksum error IPHERR = 1, FRMT = 1 and PCERR = 1: both header and payload checksum errorsThis bit is logical ORed by the following bits when DFM is equal to 1: IPPLDERR: IP frame payload error IPHERR: IP frame header error DERR: Descriptor error OERR: Overflow error LCO: Late collision RWDT: Watchdog timeout RERR: Receive error CERR: CRC error
14	DERR	Descriptor error bit Only when the LDES bit is set, this bit is valid. 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 Only when the FRMT bit is reset, this bit is valid. This bit shows whether the length field in received is mismatch 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 shows whether 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 shows whether 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 / TSV	IP frame header error bit / Timestamp valid bit When DFM=0, bit 7, 5 and 0 indicate some special cases refer to the error status table. When DFM=1, this bit indicates the timestamp value is taken and write to the Receive Descriptor6 and Receive Descriptor7. This bit is valid only when LDES is set.
6	LCO	Late collision bit This bit shows whether 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 When DFM=0, bit 7, 5 and 0 shows some special cases refer to the error status table. When DFM=1, this bit shows the received frame is an Ethernet type frame or a tagged frame. If the received frame is runt frame, this bit is not valid for application. 0: The received frame is an IEEE802.3 frame without tagged. 1: The received frame is an Ethernet-type frame (the length / type field is greater than or equal to 0x0600, or this is a tagged frame)
4	RWDT	Receive watchdog timeout bit When WDD=0, this bit shows a frame with more than 2048 bytes was detected. When WDD=1, this bit shows 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 shows whether 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 shows whether there is an incomplete byte (odd cycles during reception) received. Only when in MII interface mode, this bit is valid.

0: No dribble bit error occurred

1: Dribble bit error occurred

1 CERR

CRC error bit

This bit shows whether FCS field in received frame is mismatch with the calculation result of the hardware. Only when LDES bit is set, this bit is valid.

0: No CRC error occurred

1:A CRC error occurred

0 PCERR / EXSV

Payload checksum error bit / Extended status valid bit

When DFM=0, bit 7, 5 and 0 indicate some special cases refer to the error status table.

When DFM=1, this bit indicates the descriptor Receive Descriptor4is valid for application.

This bit only valid when LDES is set.

0: Receive Descriptor4is not valid for application

1: Receive Descriptor4is valid for application

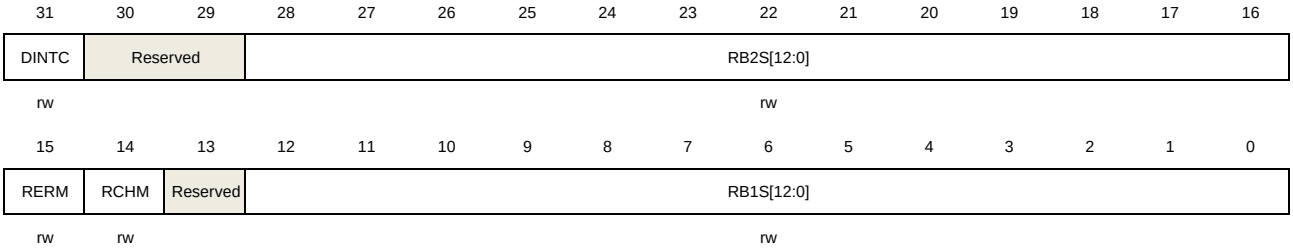
Table 27-9. Error status decoding in Receive Descriptor0, only used for normal descriptor (DFM=0) shows the combination meaning for bit 7, 5, and 0 in Receive Descriptor0:

Table 27-9. Error status decoding in Receive Descriptor0, only used for normal descriptor (DFM=0)

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 case 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 a header checksum error detected

			This error may case 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

- Receive descriptor 1



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 is not immediately set after receiving the completed, but will set after a configurable delay time.
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 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[12:0] 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.

■ Receive descriptor 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 snapshot condition. If the received frame does not meet the snapshoot condition, these bits will keep RB1AP value.

■ Receive descriptor 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															

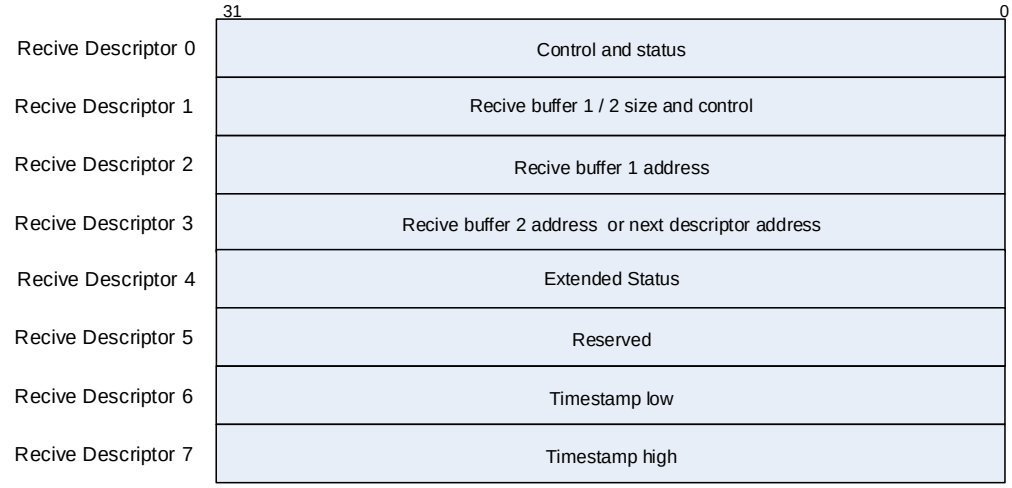
Bits	Fields	Descriptions
31:0	RB2AP / RTSH[31:0]	Receive buffer 2 address pointer (next descriptor address) / Receive frame timestamp high 32-bit value bits These bits are designed for two different functions: buffer address pointer or next descriptor address (RB1AP) or timestamp high 32-bit value (RTSH). 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. 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 snapshot condition, these bits will keep RB2AP value.

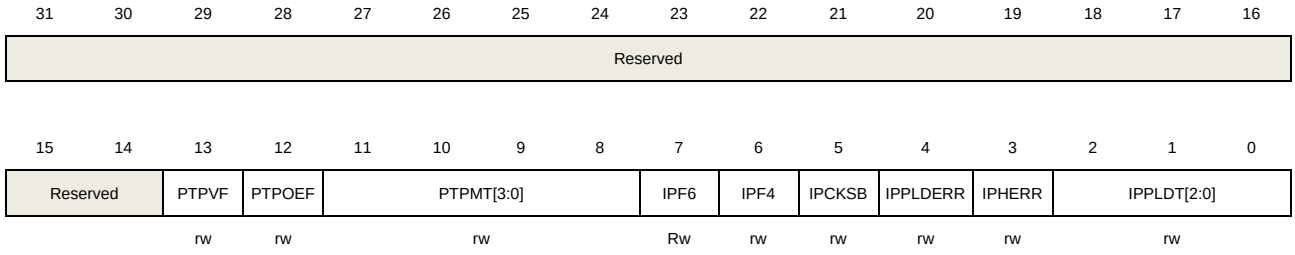
RxDMA descriptors in enhanced mode

In enhanced descriptor mode, the descriptor structure consists of eight 32-bit words: Receive Descriptor0 ~ Receive Descriptor7. The description of Receive Descriptor0 ~ Receive Descriptor3 are the same with descriptors in normal mode. The description of Receive Descriptor4 ~ Receive Descriptor7 are given below.

Figure 27-10. Receive descriptor in enhanced mode



■ Receive descriptor 4



Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13	PTPVF	PTP version format bit 0: Version 1 format 1: Version 2 format
12	PTPOEF	PTP on Ethernet frame bit 0: Received PTP frame is a IP-UDP frame if PTPMT is not zero 1: Received PTP frame is a IEEE802.3 Ethernet frame
11:8	PTPMT[3:0]	PTP message type bits PTP message type is decoded to following number:

		0x0: Not PTP frame received 0x1: SYNC 0x2: FOLLOW_UP 0x3: DELAY_REQ 0x4: DELAY_RESP 0x5: For peer-to-peer transparent clock: PDELAY_REQ For ordinary or boundary clock: ANNOUNCE 0x6: For peer-to-peer transparent clock: PDELAY_RESP For ordinary or boundary clock: MANAGEMENT 0x7: For peer-to-peer transparent clock: PDELAY_RESP_FOLLOW_UP For ordinary or boundary clock: SIGNALING
7	IPF6	IP frame in version 6 bit 0: Received frame is not a IPv6 frame 1: Received frame is a IPv6 frame
6	IPF4	IP frame in version 4 bit 0: Received frame is not a IPv4 frame 1: Received frame is a IPv4 frame
5	IPCKSB	IP frame checksum bypassed bit This bit is only valid when received frame is a IPv4 or IPv6 frame. 0: Received frame checksum checking function is not bypassed 1: Received frame checksum checking function is bypassed
4	IPPLDERR	IP frame payload error bit This bit can be set by any of below cases: 1) the calculated checksum by hardware mismatch with the TCP, UDP or ICMP checksum field in frame. 2) payload length value in IP header mismatch the received payload length. 0: Payload error not occurred in received frame 1: Payload error occurred in received frame
3	IPHERR	IP frame header error bit This bit can be set by any of below cases: 1) the calculated checksum by hardware mismatch with the IP header checksum field value. 2) Type field in IP frame is not consistent with version field (e.g. ‘type’ field value is 0x0800 but ‘version’ field value is not 0x4, ‘type’ field value is 0x86dd but ‘version’ field value is not 0x6). 0: IP header error not occurred 1: IP header error occurred
2:0	IPPLDT[2:0]	IP frame payload type bits These bits are valid only when IPFCO=1, IPHERR=0 and LDES=1. 0x0: Unsupported payload type or IP payload bypassed 0x1: payload type is UDP 0x2: payload type is TCP 0x3: payload type is ICMP

0x4~0x7: Reserved

- Receive descriptor 5

  All bits reserved

- Receive descriptor 6

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

Bits	Fields	Descriptions
31:0	RTSL[31:0]	Receive frame timestamp low 32-bit value When timestamp function is enabled and LDES is set, these bits will be written to timestamp low 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshoot condition.

- Receive descriptor 7

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

Bits	Fields	Descriptions
31:0	RTSH[31:0]	Receive frame timestamp high 32-bit value When timestamp function is enabled and LDES is set, these bits will be written to timestamp high 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshoot condition.

27.3.4. 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 definition***', there is a detailed description of the function of these registers.

- When the transmit frame does not appear the situations, such as frame underflow, no carrier, carrier lost, excessive deferral, late collision， excessive collision and jabber timeout, it can be called “good frame”. MSC transmit counters will automatically update.

- When the receiving frame does not appear the situations, such as alignment error, CRC mismatch , runt frame, length error, range error and error signal valid on pin MII_RX_ER, it can be called ‘good frame’ and MSC reception counters will automatically update. Among them, CRC mismatch indicates that calculated CRC value is different from FSC field value, runt frame indicates that the frame length is shorter than 64 bytes, length error indicates that the length field value is different from the actual received data bytes, range error indicates that the length field value is larger than maximum size of defined in IEEE802.3 (1518 for untagged frame and 1522 for VLAN tagged frame).

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.

27.3.5. 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.

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.

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 27-11. Wakeup frame filter register

Wakeup Frame Filter Register 0	Filter 0 Byte Mask							
Wakeup Frame Filter Register 1	Filter 1 Byte Mask							
Wakeup Frame Filter Register 2	Filter 2 Byte Mask							
Wakeup Frame Filter Register 3	Filter 3 Byte Mask							
Wakeup Frame Filter Register 4	Reserve	Filter 3 Command	Reserve	Filter 2 Command	Reserve	Filter 1 Command	Reserve	Filter 0 Command
Wakeup Frame Filter Register 5	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset	
Wakeup Frame Filter Register 6	Filter 1 CRC - 16				Filter 0 CRC - 16			
Wakeup Frame Filter Register 7	Filter 3 CRC - 16				Filter 2 CRC - 16			

- 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.

27.3.6. 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:

- PTP reference clock input period.
- Characteristics of the oscillator (drift).
- Frequency of the synchronization procedure.

System time calibration

PTP input reference clock is used to update 64-bit PTP system time. 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.

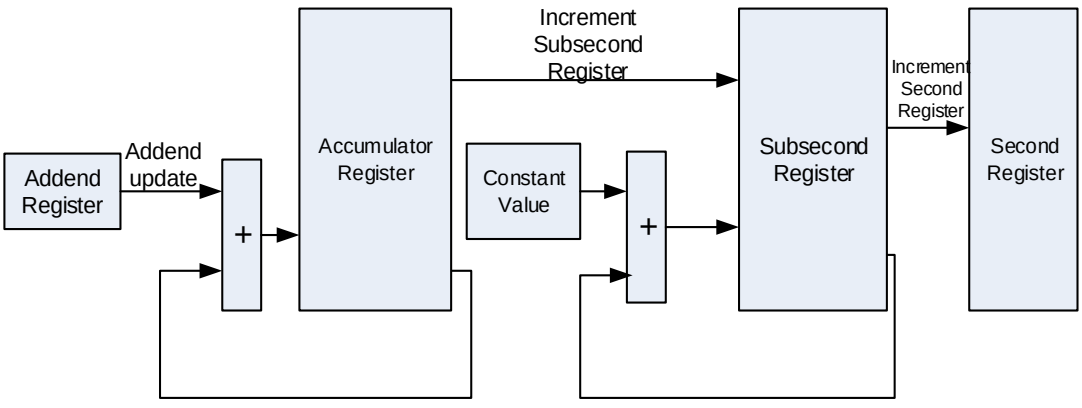
If the coarse correction method is selected, application can configure PTP timestamp update register (ENET_PTP_TSUH and ENET_PTP_TSUL) for system time initialization or correction. If TMSSTI bit is set, PTP timestamp update register is used for initialization and if TMSSTU bit is set, PTP timestamp update 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.

Figure 27-12. System time update using the fine correction method shows the fine correction algorithm process:

Figure 27-12. 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 25ns, which means the frequency of update is 40MHz. If the reference clock of HCLK is 72MHz, the frequency ratio is calculated as 72 / 40, result is 1.8. Hence, the addend (TMSA bit in ENET_PTP_TSADDEND register) value to be set is 2^{32} / 1.8, which is equal to 0x8E38 E38E. If the reference clock frequency drifts lower, for example, down to 68MHz, the frequency ratio

changes to 68 / 40 = 1.7, the value to be set in the addend register is 2^{32} / 1.7 = 0x9696 9697. If the reference clock drift higher, for example, up to 76MHz, the value should to be set in the ENET_PTP_TSADDEND register is 2^{32} / 1.9 = 0x86BC A1AF. 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 25ns. The value of the register is to update values of timestamp low 32-bit register after accumulator register overflow. Because the STMSS[30:0] bits in the ENET_PTP_TSL register 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 25ns, subsecond increment register value should be set to 25 / 0.46 = 0d54.

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 TMSTIM in the ENET_MAC_INTMSK register to mask the timestamp trigger interrupt;
2. Setting TMSEN 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 TMSARU 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 TMSARU in the ENET_PTP_TSCTL register until it is cleared;
6. Set TMSFCU 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 TMSSTI in the ENET_PTP_TSCTL register;
9. The timestamp counter starts counting as soon as the initialization process complete.

System time update steps

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.

Fine correction method

1. Calculate the value of the desired system clock rate corresponding to the addend register (***System time calibration*** has explained before);
2. Program the addend register, and set the TMSARU in ENET_PTP_TSCTL register;
3. Program the target high and low register and reset the TMSTIM of the ENET_MAC_INTMSK register to allow time stamp interrupt;
4. Set 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 TMSARU 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, which is unrelated with whether the transmitted frame has PTP feature or not, and the timestamp of all received frames will be recorded if ARFSEN bit in ENET_PTP_TSCTL register is set. If ARFSEN is reset, the received frame which passed the address filer should be matched with the configuration in ENET_PTP_TSCTL register. In another word, only the frame matched the PTP configuration is marked a PTP frame, and timestamp will be recorded in descriptor. To be marked as a PTP frame, the received frame PTP version should be coincide with PFSV bit and then the corresponding frame type enable

bit (bit 13 to bit 11 in register ENET_PTP_TSCTL) is set. Specially, the non-IP payload PTP frame (PTP on normal 802.3 Ethernet frame), also the DA should be the special MAC address (e.g. the DA should be 0x0e00 00c2 8001 for PDELAY_REQ / PDELAY_RESP / PDELAY_RESP_FOLLOW_UP message type, and the DA address 0x0000 0019 1B01 for other message type, detailed informations refer to Specification IEEE1588-2008). If MAFEN is set, this special MAC address can be extended to MAC address1-3 with SAF is reset.

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 with IEEE 1588 timestamp format***" and "***Receive DMA descriptor with IEEE 1588 timestamp format***".

Internal connection trigger

MAC can provide trigger interrupt when the system time is no less than the expected 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 expected time, the PTP module sets an output signal. Set TIMER1ITI1_REMAP bit in the AFIO_PCF0 register to 0 can make this signal internally connected to the ITI1 input of TIMER1. For this feature designed, no uncertainty is introduced because the clock of the TIMER1 and PTP reference clock (HCLK) are synchronous.

PPS output signal

Application set PTP_PPS_REMAP bit in the AFIO_PCF0 register to 1 to enable the PPS output function. This function can output a signal with the pulse width of 125ms by default (other width is detailed in ***PTP PPS control register (ENET_PTP_PPSCTL)***) 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(pulse-per-second) and connect them to one oscilloscope for clock measurement.

27.3.7. 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 11(AF11).

- **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 Transmit Descriptor;
  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 Receive Descriptor0 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.

27.3.8. 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.

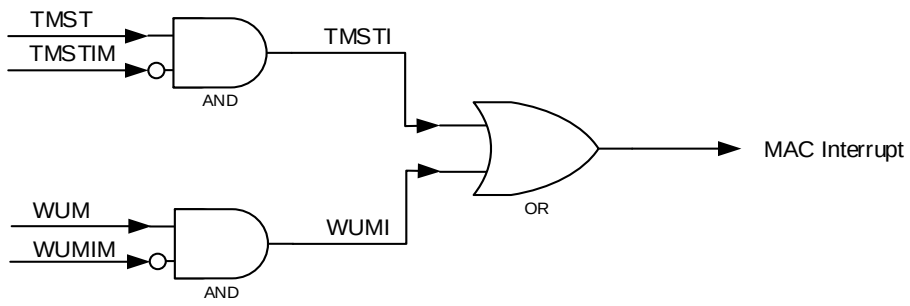
The WUM block event is connected to the second interrupt vector. The event can be remote wakeup frame received event or / and Magic Packet wakeup frame received event. This interrupt is inner mapped on the EXTI line 19. So, if the EXTI line 19 is enabled and configured to trigger by rising edge, the 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 27-13. MAC interrupt scheme



DMA controller interrupts

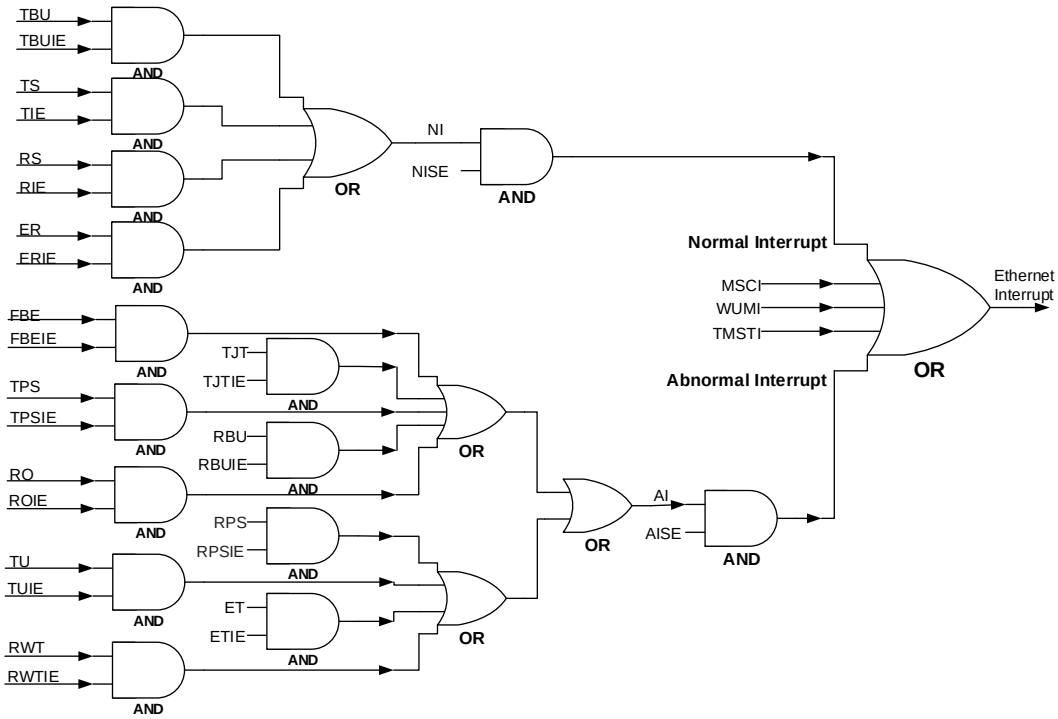
The DMA controller has two types of event: Normal and Abnormal.

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.

Figure 27-14. Ethernet interrupt scheme shows the Ethernet module interrupt connection:

Figure 27-14. Ethernet interrupt scheme



27.4. Register definition

ENET base addresss: 0x4002 8000

Byte (8-bit) access, half word (16-bit) access and word (32-bit) access are all supported for application.

27.4.1. MAC configuration register (ENET_MAC_CFG)

Address offset: 0x0000

Reset value: 0x0000 8000

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

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						TFCD	Reserved	WDD	JBD	Reserved		IGBS[2:0]			CSD
						rw		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
31:26	Reserved	Must be kept at reset value.
25	TFCD	Type Frame CRC Dropping 0: FCS field (last 4 bytes) of frame will not be dropped before forwarding 1: FCS field (last 4 bytes) of frame will be dropped before forwarding **Note:** This bit only valid when LT field of frame greater than 0x0600.
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 received frame that Less than or equals to 2048 bytes is allowed by MAC 1: The watchdog timer that on the receiver is disabled by MAC. And the received frame up to 16384 bytes is allowed by MAC.
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 carrier sense error is generated by MAC transmitter, and the transmission will be aborted. 1: The MII CRS signal is ignored by MAC transmitter while in frame transmitting. 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 When in Full-duplex mode, this bit can be ignored. 0: The packets that transmitting from PHY are all received by MAC 1: Receiving frames from PHY is disabled by MAC
12	LBM	Loopback mode bit 0: The MAC is configured in normal mode 1: The MAC is configured 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 When in Full-duplex mode, this bit can be ignored. 0: Up to 16 times retries based on the settings of BOL is attempted by MAC 1: Only 1 transmission is attempted by MAC
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 in Full-duplex mode, these bits can be ignored.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: $0 \leq dt < 2^{k}$. 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
4	DFC	Deferral check bit When in Full-duplex mode, this bit can be ignored. 0: Disable the deferral check function of MAC. Until the CRS signals changed to inactive, the MAC defers sending. 1: Enable the deferral check function of 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.
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.

27.4.2. MAC frame filter register (ENET_MAC_FRMF)

Address offset: 0x0004
Reset value: 0x0000 0000

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

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 received 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: The MAC does not forward any control frames to the application 0x1: The MAC forwards any control frames except pause control frames to the application 0x2: Even if the control frames failed the address filter, the MAC forwards all of them to application 0x3: Only the control frames pass the address filter, the MAC forwards them to application
5	BFRMD	Broadcast frames disable bit

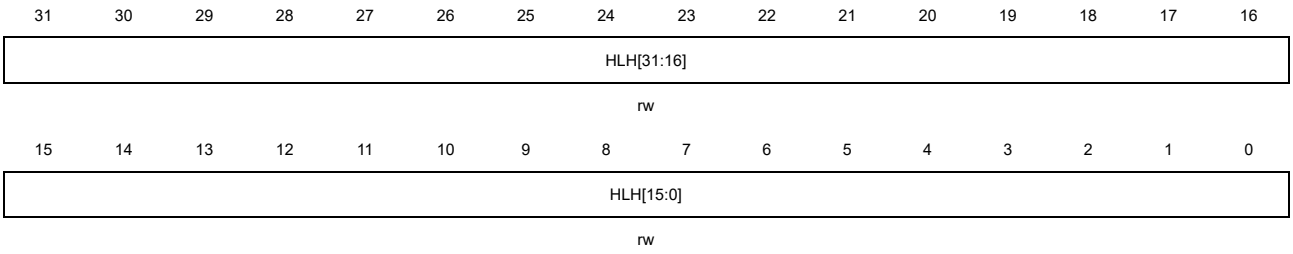
		0: Ignore the address filters, and all received broadcast frames is passed. 1: All received broadcast frames is filtered by address filters
4	MFD	Multicast filter disable bit 0: Multicast filter is enabled. The filtering mode of multicast frame is determined by HMF bit. 1: Multicast filter is disabled. All received multicast frames are passed. The first bit in the destination address field of multicast frames is '1', but not all bits in the destination are '1'.
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

27.4.3. MAC hash list high register (ENET_MAC_HLH)

Address offset: 0x0008

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:0	HLH[31:0]	Hash list high bits These bits take the high 32-bit value of hash list.

27.4.4. MAC hash list low register (ENET_MAC_HLL)

Address offset: 0x000C

Reset value: 0x0000 0000

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

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.

27.4.5. MAC PHY control register (ENET_MAC_PHY_CTL)

Address offset: 0x0010

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), 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
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: 60-100 MHz) 0x1: HCLK / 62 (HCLK range: 100-150 MHz)

		0x2: HCLK / 16 (HCLK range: 20-35 MHz) 0x3: HCLK / 26 (HCLK range: 35-60 MHz) 0x4: HCLK / 102 (HCLK range: 150-180 MHz) other: Reserved
1	PW	PHY write bit This bit indicates 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.

27.4.6. MAC PHY data register (ENET_MAC_PHY_DATA)

Address offset: 0x0014
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), 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
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.

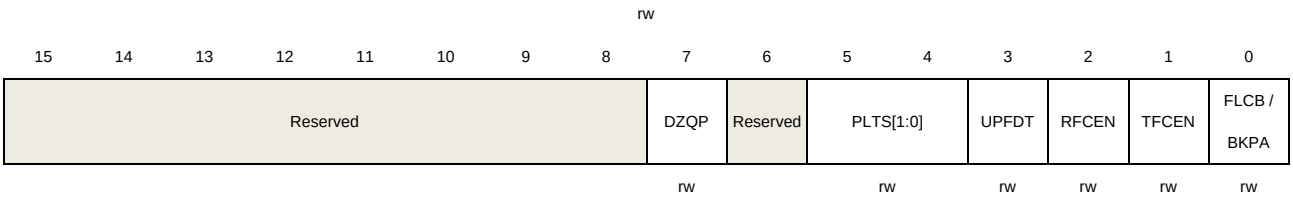
27.4.7. MAC flow control register (ENET_MAC_FCTL)

Address offset: 0x0018
Reset value: 0x0000 0000

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

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]															



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: 4 slot times is subtracted by pause time 0x1: 28 slot times is subtracted by pause time 0x2: 144 slot times is subtracted by pause time 0x3: 256 slot times is subtracted by pause time **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-

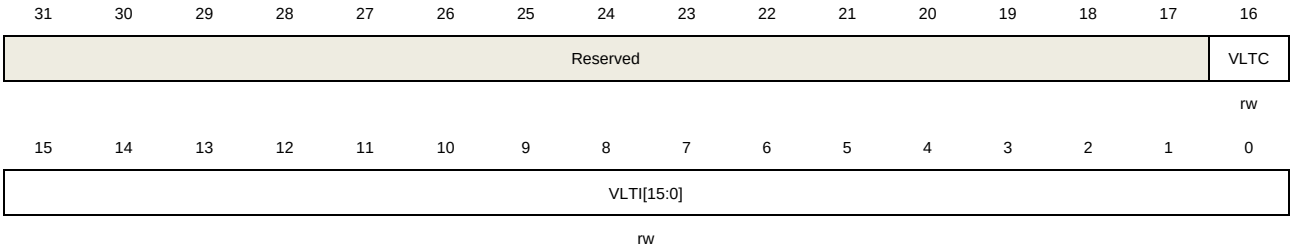
		duplex mode and back-pressure feature in Half-duplex mode can be performed by transmitter.
0	FLCB / BKPA	Flow control busy / back pressure activate bit This bit only valid when TFCEN is set. This bit can send a pause frame in Full-duplex mode or activate the back pressure function in Half-duplex mode 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.

27.4.8. MAC VLAN tag register (ENET_MAC_VLT)

Address offset: 0x001C
Reset value: 0x0000 0000

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

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.



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.

27.4.9. MAC remote wakeup frame filter register (ENET_MAC_RWFF)

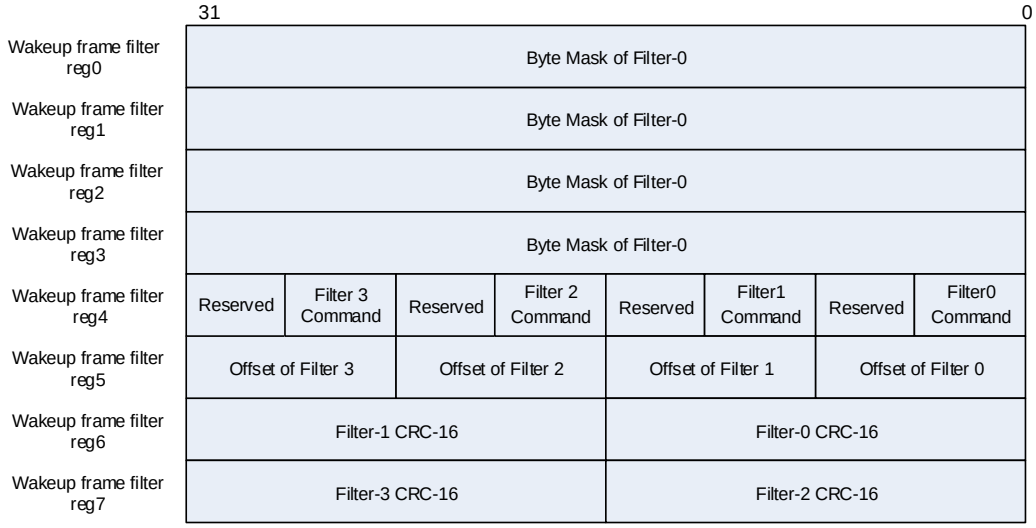
Address offset: 0x0028

Reset value: 0x0000 0000

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

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 27-15. Wakeup frame filter register



27.4.10. MAC wakeup management register (ENET_MAC_WUM)

Address offset: 0x002C

Reset value: 0x0000 0000

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

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: Received the Magic Packet frame, and generating the wakeup event
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.

27.4.11. MAC debug register (ENET_MAC_DBG)

Address offset: 0x0034

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), 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						TXFF	TXFNE	Reserved	TXFW	TXFRS[1:0]		PCS	SOMT[1:0]		MTNI
						ro	ro		ro	ro		ro	ro		ro

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						RXFS[1:0]		Reserved	RXFRS[1:0]		RXFW	Reserved	RXAFS[1:0]		MRNI
						ro			ro		ro		ro		ro

Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value.
25	TXFF	TxFIFO Full flag 0: TxFIFO is not full 1: TxFIFO is full
24	TXFNE	TxFIFO not empty flag 0: TxFIFO is empty 1: TxFIFO is not empty
23	Reserved	Must be kept at reset value.
22	TXFW	TxFIFO is writing 0: TxFIFO is not doing write operation 1: TxFIFO is doing write operation
21:20	TXFRS[1:0]	TxFIFO read operation status 0x0: TxFIFO read controller is in idle state 0x1: TxFIFO read controller is in reading state 0x2: TxFIFO read controller is in waiting feedback Tx status from MAC transmitter 0x3: TxFIFO read controller is in writing the Tx descriptor status or flushing the TxFIFO
19	PCS	Pause condition status 0: MAC transmitter is not in pause condition 1: MAC transmitter is under pause condition and will delay transmitting frame
18:17	SOMT[1:0]	Status of MAC transmitter 0x0: The MAC transmitter controller is in idle state 0x1: The MAC transmitter controller is in Waiting feedback of previous frame status or the end of IFG / BACKOFF period 0x2: For Full-duplex mode, indicates pause control frame is transmitting 0x3: The MAC transmitter controller is in Reading input frame from FIFO for transmission
16	MTNI	MAC transmit state not idle 0: MAC transmitter is in idle state 1: MAC transmitter is not in idle state

15:10	Reserved	Must be kept at reset value.
9:8	RXFS	RxFIFO state 0x0: The RxFIFO is empty 0x1: The flow-control low threshold is greater than RxFIFO number of value 0x2: The flow-control high threshold is lower than RxFIFO number of value 0x3: The RxFIFO is full
7	Reserved	Must be kept at reset value.
6:5	RXFRS[1:0]	RxFIFO read operation status 0x0: RxFIFO read controller is in idle state 0x1: RxFIFO read controller is in reading state 0x2: RxFIFO read controller is reading frame status (including time-stamp) 0x3: RxFIFO read controller is flushing frame
4	RXFW	RxFIFO is writing 0: RxFIFO is not doing write operation 1: RxFIFO is doing write operation
3	Reserved	Must be kept at reset value.
2:1	RXAFS[1:0]	Rx asynchronous FIFO status RXAFS[1]:Rx asynchronous FIFO reading state in HCLK Clock domain RXAFS[0]:Rx asynchronous FIFO writing state in MAC RX_CLK Clock domain
0	MRNI	MAC receive state not idle 0: MAC receiver is in idle state 1: MAC receiver is not in idle state

27.4.12. MAC interrupt flag register (ENET_MAC_INTF)

Address offset: 0x0038

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), 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						TMST	Reserved		MSCT	MSCR	MSC	WUM	Reserved		
						rc_r			r	r	r	r			

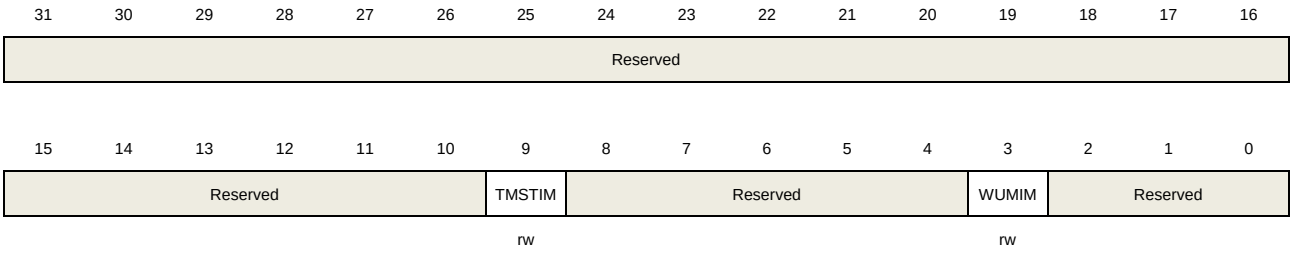
Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	TMST	Time stamp trigger status bit

		This bit is cleared when ENET_PTP_TSF register is read. 0: The system time value is less than the value specified in the the ENET_PTP_ETH and ENET_PTP_ETL registers 1: The system time value is no less than the value specified in the ENET_PTP_ETH and ENET_PTP_ETL 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.

27.4.13. MAC interrupt mask register (ENET_MAC_INTMSK)

Address offset: 0x003C
Reset value: 0x0000 0000

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



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.

27.4.14. MAC address 0 high register (ENET_MAC_ADDR0H)

Address offset: 0x0040
Reset value: 0x8000 FFFF

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

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.

27.4.15. MAC address 0 low register (ENET_MAC_ADDR0L)

Address offset: 0x0044
Reset value: 0xFFFF FFFF

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

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.

27.4.16. MAC address 1 high register (ENET_MAC_ADDR1H)

Address offset: 0x0048

Reset value: 0x0000 FFFF

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

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: MAC address1 is ignored by address filter for filtering 1: MAC address1 is used by address filter for perfect filtering
30	SAF	Source address filter bit 0: Comparing MAC address1 with the destination address field of the received frame 1: Comparing MAC address1 with the source address field of the received frame
29:24	MB[5:0]	Mask byte bits If these bits is set, the destination address / source address corresponding byte of the received frame is not compared with MAC address1. 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.

27.4.17. MAC address 1 low register (ENET_MAC_ADDR1L)

Address offset: 0x004C

Reset value: 0xFFFF FFFF

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

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.

27.4.18. MAC address 2 high register (ENET_MAC_ADDR2H)

Address offset: 0x0050

Reset value: 0x0000 FFFF

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

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: MAC address2 is ignored by address filter for filtering 1: MAC address2 is used by address filter for perfect filtering
30	SAF	Source address filter bit 0: Comparing MAC address2 with the destination address field of the received frame 1: Comparing MAC address2 with the source address field of the received frame
29:24	MB[5:0]	Mask byte bits If these bits is set, the destination address / source address corresponding byte of the received frame is not compared with MAC address2. Each bit controls one byte mask as follows:

MB[5]: ENET_MAC_ADDR2H[15:8]
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.

27.4.19. MAC address 2 low register (ENET_MAC_ADDR2L)

Address offset: 0x0054
Reset value: 0xFFFF FFFF

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR2L[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR2L[15:0]															
rw															

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.

27.4.20. MAC address 3 high register (ENET_MAC_ADDR3H)

Address offset: 0x0058
Reset value: 0x0000 FFFF

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

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
ADDR3H[15:0]															
rw															

Bits	Fields	Descriptions
31	AFE	Address filter enable bit

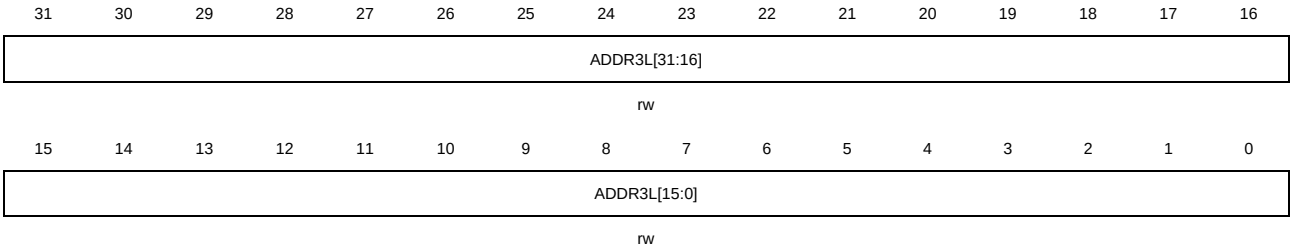
		0: MAC address3 is ignored by address filter for filtering 1: MAC address3 is used by address filter for perfect filtering
30	SAF	Source address filter bit 0: Comparing MAC address3 with the destination address field of the received frame 1: Comparing MAC address3 with the source address field of the received frame
29:24	MB[5:0]	Mask byte bits If these bits is set, the destination address / source address corresponding byte of the received frame is not compared with MAC address3. 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.

27.4.21. MAC address 3 low register (ENET_MAC_ADDR3L)

Address offset: 0x005C

Reset value: 0xFFFF FFFF

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



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.

27.4.22. MAC flow control threshold register (ENET_MAC_FCTH)

Address offset: 0x1080

Reset value: 0x0000 0015

This register can be accessed by byte (8-bit), 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									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

27.4.23. MSC control register (ENET_MSC_CTL)

Address offset: 0x0100
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), 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										AFHPM	PMC	MCFZ	RTOR	CTSR	CTR
										rw	wo	rw	rw	rw	rw

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5	AFHPM	Almost full or half preset mode 0: Preset all MSC counters to almost-half (0x7FFF FFF0) value 1: Preset all MSC counters to almost-full (0xFFFF FFF0) value **Note:** This bit is valid only when PMC is set.
4	PMC	Preset MSC counter 0: No effect 1: Preset MSC counters to a preset value. Preset value depends on AFHPM.
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

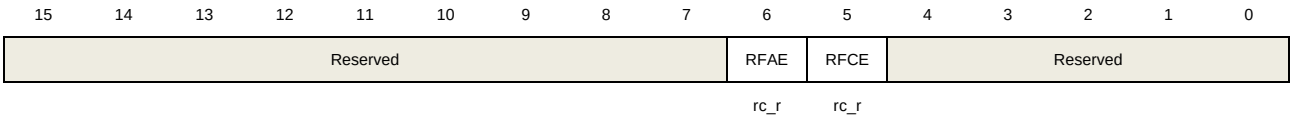
27.4.24. MSC receive interrupt flag register (ENET_MSC_RINTF)

Address offset: 0x0104

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), 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														RGUF	Reserved
														rc_r	



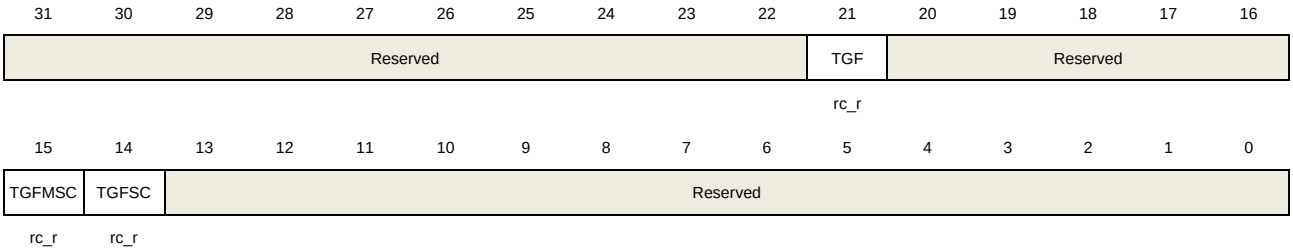
Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.
17	RGUF	Received good unicast frames bit 0: Good unicast frame received counter is less than half of the maximum value 1: Good unicast frame received counter reaches half of the maximum value
16:7	Reserved	Must be kept at reset value.
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.

27.4.25. MSC transmit interrupt flag register (ENET_MSC_TINTF)

Address offset: 0x0108

Reset value: 0x0000 0000

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



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

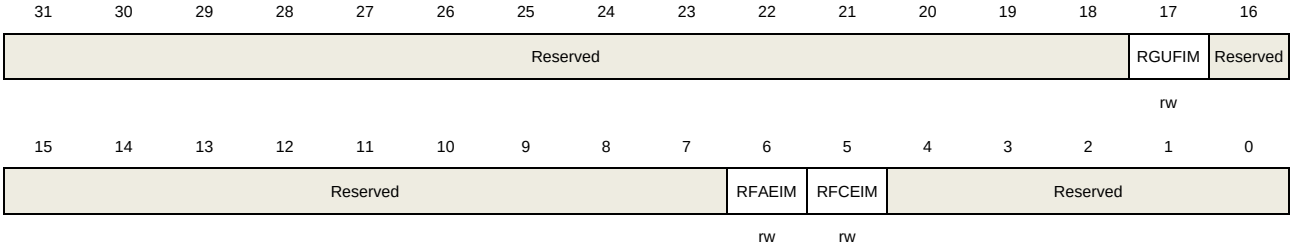
		of the maximum value 1:Good frame after more than a single collision transmitted counter reaches half of the maximum value
14	TGFSC	Transmitted good frames single collision bit 0: Good frame after a single collision transmitted counter is less than half of the maximum value 1: Good frame after a single collision transmitted counter reaches half of the maximum value
13:0	Reserved	Must be kept at reset value.

27.4.26. MSC receive interrupt mask register (ENET_MSC_RINTMSK)

Address offset: 0x010C
Reset value: 0x0000 0000

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

The Ethernet MSC receive interrupt mask register maintains the masks for interrupts generated when receive statistic counters reach half their maximum value



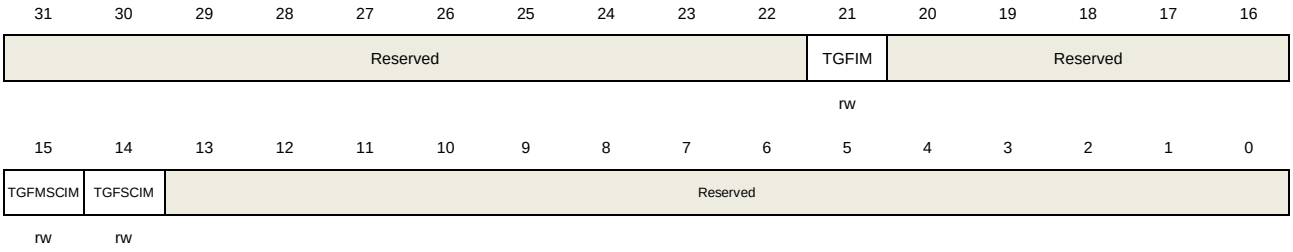
Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.
17	RGUFIM	Received good unicast frames interrupt mask bit 0: Unmask the interrupt when the RGUF bit is set 1: Mask the interrupt when RGUF bit is set
16:7	Reserved	Must be kept at reset value.
6	RFAEIM	Received frames alignment error 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.

27.4.27. MSC transmit interrupt mask register (ENET_MSC_TINTMSK)

Address offset: 0x0110
Reset value: 0x0000 0000

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

The MSC transmit interrupt mask register configures the mask bits for interrupts generation



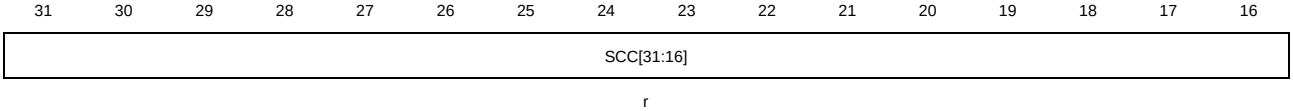
Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value.
21	TGFIM	Transmitted good frames interrupt mask bit 0: Unmask the interrupt when the TGF bit is set 1:Mask the interrupt when the TGF bit is set
20:16	Reserved	Must be kept at reset value.
15	TGFMSCIM	Transmitted good frames more single 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.

27.4.28. MSC transmitted good frames after a single collision counter register (ENET_MSC_SCCNT)

Address offset: 0x014C
Reset value: 0x0000 0000

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

This register counts the number of successfully transmitted frames after a single collision in Half-duplex mode.



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.

27.4.29. MSC transmitted good frames after more than a single collision counter register (ENET_MSC_MSCCNT)

Address offset: 0x0150
Reset value: 0x0000 0000

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

This register counts the number of successfully transmitted frames after more than one single collision in Half-duplex mode.

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

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.

27.4.30. MSC transmitted good frames counter register (ENET_MSC_TGFCNT)

Address offset: 0x0168
Reset value: 0x0000 0000

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

This register counts the number of good frames transmitted.

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

TGF[15:0]

r

Bits	Fields	Descriptions
31:0	TGF[31:0]	Transmitted good frames counter bits These bits count the number of transmitted good frames.

27.4.31. MSC received frames with CRC error counter register (ENET_MSC_RFCECNT)

Address offset: 0x0194
Reset value: 0x0000 0000

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

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.

27.4.32. MSC received frames with alignment error counter register (ENET_MSC_RFAECNT)

Address offset: 0x0198
Reset value: 0x0000 0000

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

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.

27.4.33. MSC received good unicast frames counter register (ENET_MSC_RGUFCNT)

Address offset: 0x01C4
Reset value: 0x0000 0000

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

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.

27.4.34. PTP time stamp control register (ENET_PTP_TSCTL)

Address offset: 0x0700
Reset value: 0x0000 2000

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

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													MAFEN	CKNT[1:0]	
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MNMSEN	ETMSEN	IP4SEN	IP6SEN	ESEN	PFSV	SCROM	ARFSEN	Reserved		TMSARU	TMSITEN	TMSSTU	TMSSTI	TMSFCU	TMSEN
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value.
18	MAFEN	MAC address filter enable for PTP frame 0: No effect

		1: Enable MAC address1-3 to filter the PTP frame when received frame's type field is 0x88f7
17:16	CKNT[1:0]	Clock node type for time stamp 0x0: Type of ordinary clock 0x1: Type of boundary clock 0x2: Type of end-to-end transparent clock 0x3: Type of peer-to-peer transparent clock
15	MNMSEN	Received master node message snapshot enable This bit is valid only when CKNT=0x0 or 0x1. 0: Snapshot is only taken for slave node message 1: Snapshot is only take for master node message
14	ETMSEN	Received event type message snapshot enable 0: All type messages are taken snapshot except Announce, Management and Signaling message 1: Only event type messages (SYNC, DELAY_REQ, PDELAY_REQ and PDELAY_RESP) are taken snapshot
13	IP4SEN	Received IPv4 snapshot enable 0: Do not take snapshot for IPv4 frame 1: Take snapshot for IPv4 frame
12	IP6SEN	Received IPv6 snapshot enable 0: Do not take snapshot for IPv6 frame 1: Take snapshot for IPv6 frame
11	ESEN	Received Ethernet snapshot enable 0: Do not take snapshot when received non type frame 1: Take snapshot when received non type frame
10	PFSV	PTP frame snooping version 0: Version 1 (Revision of IEEE STD. 1588-2002 / 1588-2008) 1: Version 2 (Revision of IEEE STD. 1588-2008)
9	SCROM	Subsecond counter rollover mode 0: Binary rollover mode. Subsecond rollovers when reach 0x7FFF_FFFF 1: Digital rollover mode. Subsecond rollovers when reach 0x3B9A_C9FF (0d999_999_999)
8	ARFSEN	All received frames snapshot enable 0: Not all received frames are taken snapshot 1: All received frames are taken snapshot
5	TMSARU	Time stamp addend register update bit 0: The value of ENET_PTP_TSADDEND register is not updated to the PTP block for fine correction 1: The value of ENET_PTP_TSADDEND register is updated to the PTP block for

fine correction

Note: Before user set it, the TMSARU bit must be read as 0. When update is finish, the TMSARU bit is cleared.

4	TMSITEN	Timestamp interrupt trigger enable bit 0: Disable timestamp interrupt 1: When the system time is no less than the value in ENET_PTP_ETH and ENET_PTP_ETL registers, a timestamp interrupt is generated. **Note**: After the timestamp trigger interrupt happened the TMSITEN 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: Not update the system time 1: Update the system time with the value in the ENET_PTP_TSUH and ENET_PTP_TSUL registers. It is cleared by hardware when the update finished.
2	TMSSTI	Timestamp system time initialize bit This bit must be read as 0 before application set it. 0: The system time is maintained without any change 1: Initializing the system time with the value in ENET_PTP_TSUH and ENET_PTP_TSUL 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.

Table 27-10. Supported time stamp snapshot with PTP register configuration

CKNT (Bit 17:16)	**0X**			**10**		**11**	
MNMSEN (Bit 15)	X(*)	1	0	X			
ETMSEN (Bit 14)	0	1	1	0	1	0	1
Supported message type for snapshot	SYNC FOLLOW_UP DELAY_REQ	DELAY_REQ	SYNC	SYNC FOLLOW_UP DELAY_REQ	SYNC FOLLOW_UP	SYNC FOLLOW_UP DELAY_REQ DELAY_RESP PDELAY_REQ	SYNC PDELAY_REQ PDELAY_RESP

	DELAY_RESP			DELAY_RESP		PDELAY_RESP	

*: means do not care

27.4.35. PTP subsecond increment register (ENET_PTP_SSINC)

Address offset: 0x0704
Reset value: 0x0000 0000

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

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.

27.4.36. PTP time stamp high register (ENET_PTP_TSH)

Address offset: 0x0708
Reset value: 0x0000 0000

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

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.

27.4.37. PTP time stamp low register (ENET_PTP_TSL)

Address offset: 0x070C
Reset value: 0x0000 0000

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

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

27.4.38. PTP time stamp update high register (ENET_PTP_TSUH)

Address offset: 0x0710
Reset value: 0x0000 0000

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

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.

27.4.39. PTP time stamp update low register (ENET_PTP_TSUL)

Address offset: 0x0714
Reset value: 0x0000 0000

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

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.

27.4.40. PTP time stamp addend register (ENET_PTP_TSADDEND)

Address offset: 0x0718
Reset value: 0x0000 0000

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

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 In order to achieve time synchronization, the value of TMSA[31:0] is added to the accumulator register.

27.4.41. PTP expected time high register (ENET_PTP_ETH)

Address offset: 0x071C

Reset value: 0x0000 0000

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

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.

27.4.42. PTP expected time low register (ENET_PTP_ETL)

Address offset: 0x0720

Reset value: 0x0000 0000

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

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).

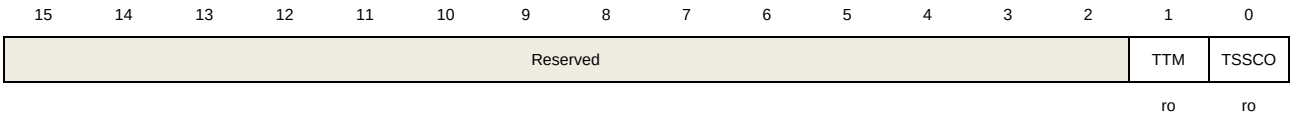
27.4.43. PTP time stamp flag register (ENET_PTP_TSF)

Address offset: 0x0728

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), 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															



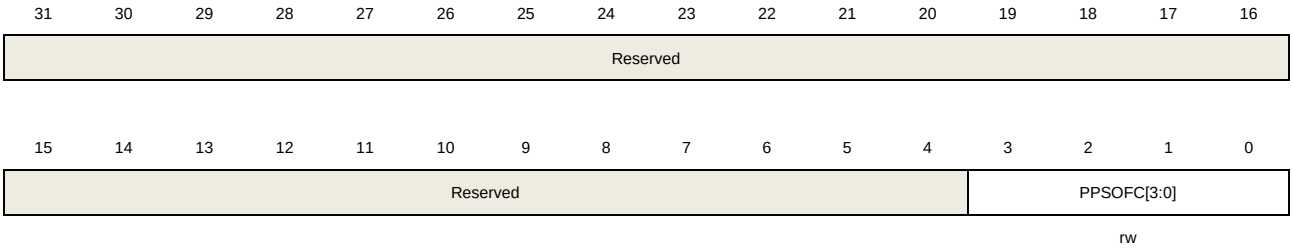
Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	TTM	Target time match bit 0: System time is not equal or greater than expected time. 1: System time is equal or greater than expected time **Note:** Reading ENET_PTP_TSF register will clear this bit.
0	TSSCO	Timestamp second counter overflow bit 0: Timestamp second counter has not overflowed 1: Timestamp second counter is greater than 0xFFFF FFFF

27.4.44. PTP PPS control register (ENET_PTP_PPSCTL)

Address offset: 0x072C

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.
3:0	PPSOFC	PPS output frequency configure 0x0: 1Hz (Pulse width: 125ms for binary rollover, 100ms for digital rollover) 0x1: 2Hz (Pulse width: 50% duty cycle for binary rollover) 0x2: 4Hz (Pulse width: 50% duty cycle for binary rollover) …. 0xF: 32768 (2^{15}) Hz (Pulse width: 50% duty cycle for binary rollover) **Note:** If digital rollover is selected, only PPSOFC=0 is recommended.

27.4.45. DMA bus control register (ENET_DMA_BCTL)

Address offset: 0x1000

Reset value: 0x0002 0101

This register can be accessed by byte (8-bit), 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					MB	AA	FPBL	UIP	RXDP[5:0]						FB
					rw	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]						DFM	DPSL[4:0]					DAB	SWR
rw		rw						rw	rw					rw	rs

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26	MB	Mixed burst 0: AHB master interface only transfer fixed burst length with 16 and below 1: AHB master interface will transfer burst length greater than 16 with INCR **Note:** MB and FB should be and must be only one of bit is set.
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: Both SINGLE and INCR burst transfer operations can be used by AHB 1: Only SINGLE, INCR4, INCR8 or INCR16 can be used by AHB, while in the start of normal burst transfer. **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	DFM	Descriptor format mode 0: Normal mode descriptor 1: Enhanced mode descriptor
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 ring descriptors. DPSL[4:0] represents the address difference from the end of the current descriptor to the beginning of the next descriptor. If the value of DPSL[4:0] is 0, the DMA taking the descriptor table as contiguous.
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

27.4.46. DMA transmit poll enable register (ENET_DMA_TPEN)

Address offset: 0x1004
Reset value: 0x0000 0000

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

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.

27.4.47. DMA receive poll enable register (ENET_DMA_RPEN)

Address offset: 0x1008
Reset value: 0x0000 0000

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

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.

27.4.48. DMA receive descriptor table address register (ENET_DMA_RDTADDR)

Address offset: 0x100C
Reset value: 0x0000 0000

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

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.

27.4.49. DMA transmit descriptor table address register (ENET_DMA_TDTADDR)

Address offset: 0x1010
Reset value: 0x0000 0000

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

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.

27.4.50. DMA status register (ENET_DMA_STAT)

Address offset: 0x1014
Reset value: 0x0000 0000

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

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 occurs while TxDMA transfer data 0: Error occurs while RxDMA transfer data EB[1]: 1: Error occurs while read transfer 0: Error occurs while write transfer EB[2]: 1: Error occurs while access descriptor 0: Error occurs while access data buffer
22:20	TP[2:0]	Transmit process state bit These bits decode the TxDMA state. 0x0: Stopped; Issuing transmit command which is Reset or Stop. 0x1: Running; Fetching the transfer descriptor that belongs to transmit. 0x2: Running; Waiting for status 0x3: Running; Queuing it to TxFIFO after reading transmit packet data from host memory buffer. 0x4, 0x5: Reserved 0x6: Suspended; Unavailable of transmit descriptor or underflow of transmit buffer. 0x7: Running; Closing the descriptor that belongs to transmit.
19:17	RP[2:0]	Receive process state bit These bits decode the RxDMA state. 0x0: Stopped; Issuing receive command which is Reset or Stop. 0x1: Running; Fetching the transfer descriptor that belongs to receive. 0x2: Reserved 0x3: Running; Waiting for the packet that belongs to receive. 0x4: Suspended: Unavailable of receive descriptor 0x5: Running: Closing the descriptor that belongs to receive. 0x6: Reserved 0x7: Running; Transferring it to host memory after reading the receive packet data from RxFIFO.
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: Interrupt of transmit TBU: Unavailable of transmit buffer RS: Interrupt of receive ER: Interrupt of early receive **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: Halt of transmit process TJT: Timeout of transmit jabber RO: Overflow of receive FIFO TU: Underflow transmit RBU: Receive buffer RPS: Unavailable of receive process stopped RWT: Timeout of receive watchdog ET: Interrupt of early transmit FBE: Error of 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 RS bit 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 OERR bit 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 Transmit Descriptor0 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 Transmit Descriptor0. 0: Current frame transmission is not finished 1: Current frame transmission is finished.

27.4.51. DMA control register (ENET_DMA_CTL)

Address offset: 0x1018
Reset value: 0x0000 0000

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

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.

27.4.52. DMA interrupt enable register (ENET_DMA_INTEN)

Address offset: 0x101C

Reset value: 0x0000 0000

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

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: Disable normal interrupt 1: Enable normal interrupt This bit enables the following bits: TS: Interrupt of transmit TBU: Unavailable transmit buffer RS: Interrupt of receive ER: Interrupt of Early receive
15	AIE	Abnormal interrupt summary enable bit 0: Disable abnormal interrupt 1: Enable abnormal interrupt This bit enables the following bits: TPS: Halt of transmit process TJT: Timeout of transmit jabber RO: Overflow of receive FIFO TU: Underflow transmit RBU: Receive buffer RPS: Unavailable of receive process stopped RWT: Timeout of receive watchdog ET: Interrupt of early transmit FBE: Error of fatal bus error
14	ERIE	Early receive interrupt enable bit 0: Disable early receive interrupt 1: Enable early receive interrupt
13	FBEIE	Fatal bus error interrupt enable bit 0: Disable fatal bus error interrupt 1: Enable fatal bus error interrupt
12:11	Reserved	Must be kept at reset value.
10	ETIE	Early transmit interrupt enable bit 0: Disable early transmit interrupt 1: Enable early transmit interrupt
9	RWTIE	Receive watchdog timeout interrupt enable bit 0: Disable receive watchdog timeout interrupt 1: Enable receive watchdog timeout interrupt
8	RPSIE	Receive process stopped interrupt enable bit 0: Disable receive stopped interrupt 1: Enable receive stopped interrupt

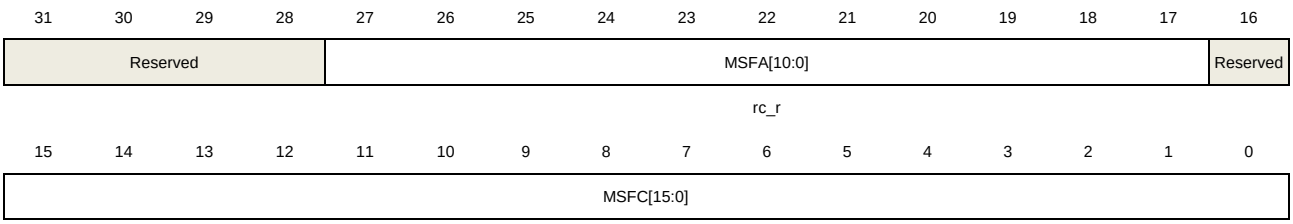
7	RBUIE	Receive buffer unavailable interrupt enable bit 0: Disable receive buffer unavailable interrupt 1: Enable receive buffer unavailable interrupt
6	RIE	Receive interrupt enable bit 0: Disable receive interrupt 1: Enable receive interrupt
5	TUIE	Transmit underflow interrupt enable bit 0: Disable underflow interrupt 1: Enable underflow interrupt
4	ROIE	Receive overflow interrupt enable bit 0: Disable overflow interrupt 1: Enable overflow interrupt
3	TJTIE	Transmit jabber timeout interrupt enable bit 0: Disable transmit jabber timeout interrupt 1: Enable transmit jabber timeout interrupt
2	TBUIE	Transmit buffer unavailable interrupt enable bit 0: Disable transmit buffer unavailable interrupt 1: Enable transmit buffer unavailable interrupt
1	TPSIE	Transmit process stopped interrupt enable bit 0: Disable transmission stopped interrupt 1: Enable transmission stopped interrupt
0	TIE	Transmit interrupt enable bit 0: Disable transmit interrupt 1: Enable transmit interrupt

27.4.53. DMA missed frame and buffer overflow counter register (ENET_DMA_MFBOCNT)

Address offset: 0x1020
Reset value: 0x0000 0000

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

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.



rc_r

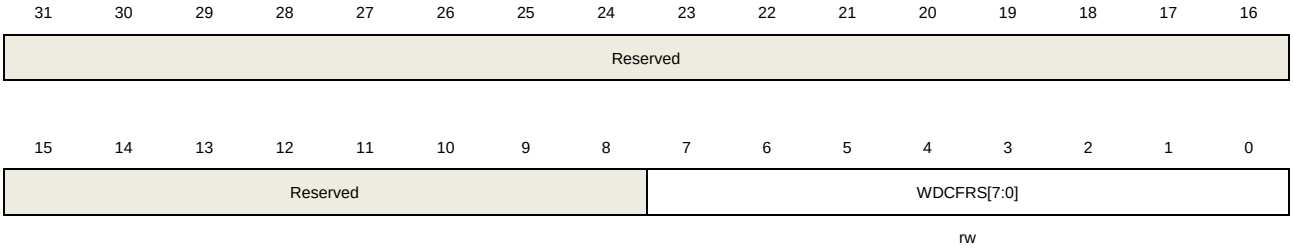
Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:17	MSFA[10:0]	Missed frames by the application bits These bits indicate the number of frames dropped by RxFIFO.
16	Reserved	Must be kept at reset value.
15:0	MSFC[15:0]	Missed frames by the controller bits These bits indicate the 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.

27.4.54. DMA receive state watchdog counter register (ENET_DMA_RSWDC)

Address offset: 0x1024
Reset value: 0x0000 0000

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

The watchdog counter value register for RS bit (ENET_DMA_STAT register) set after delay a configured time.



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	WDCFRS[7:0]	Watchdog counter for receive status (RS) bit These bits are only valid when DINTC (RXDES1) is set. When DINTC=1 and a frame is received, the RS bit will be set delay a time of WDCFRS*256 HCLK after receiving complete.

27.4.55. DMA current transmit descriptor address register (ENET_DMA_CTDADDR)

Address offset: 0x1048
Reset value: 0x0000 0000

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

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.

27.4.56. DMA current receive descriptor address register (ENET_DMA_CRDADDR)

Address offset: 0x104C
Reset value: 0x0000 0000

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

This register points to the start descriptor address of the current receive descriptor read by the RxDMA controller.

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

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

Bits	Fields	Descriptions
31:0	RDAP[31:0]	Receive descriptor address pointer bits These bits are automatically updated by RxDMA controller during operation.

27.4.57. DMA current transmit buffer address register (ENET_DMA_CTBADDR)

Address offset: 0x1050
Reset value: 0x0000 0000

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

This register points to the current transmit buffer address being read by the TxDMA controller.

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

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

Bits	Fields	Descriptions
31:0	TBAP[31:0]	Transmit buffer address pointer bits These bits are automatically updated by TxDMA controller during operation.

27.4.58. DMA current receive buffer address register (ENET_DMA_CRBADDR)

Address offset: 0x1054

Reset value: 0x0000 0000

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

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.

28. Universal Serial Bus full-speed device interface (USBD)

The USBD is only available on HD series.

28.1. Overview

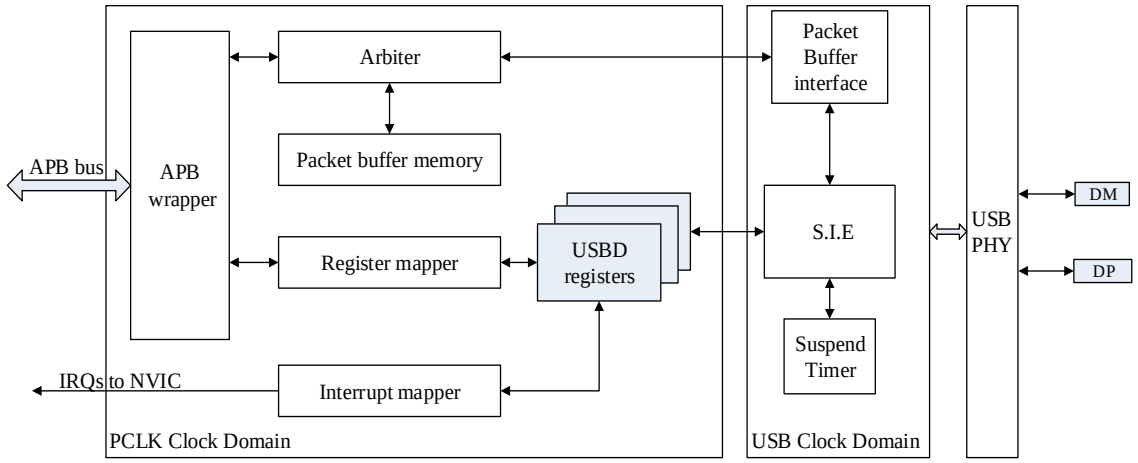
The Universal Serial Bus full-speed device interface (USBD) module provides a device solution for implementing a USB 2.0 full-speed compliant peripheral. It contains a full-speed internal USB PHY and no more external PHY chip is needed. USBD supports all the four types of transfer (control, bulk, interrupt and isochronous) defined in USB 2.0 protocol.

28.2. Main features

- USB 2.0 full-speed device controller.
- Support up to 8 configurable bidirectional endpoints.
- Support double-buffered bulk/isochronous endpoints.
- Support USB 2.0 Link Power Management.
- Each endpoint supports control, bulk, isochronous or interrupt transfer types (exclude endpoint 0, endpoint 0 only support control transfer).
- Support USB suspend/resume operations.
- Dedicated 512-byte SRAM used for data packet buffer.
- Integrated USB PHY.

28.3. Block diagram

Figure 28-1. USBD block diagram



28.4. Signal description

Table 28-1. USBD signal description

I/O port	Type	Description
DM	Input/Output	Differential data line - port
DP	Input/Output	Differential data line + port

Note: As soon as the USBD is enabled, these pins are connected to the USBD internal transceiver automatically.

28.5. 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 7 USB frequency prescaler can be used as 1, 1.5, 2, 2.5, 3, 3.5 and 4 (4 cannot be used, because the frequency of MCU cannot reach above 192MHz). Thus, for obtaining 48MHz clock, PLL frequencies doubling can only be 48MHz, 72MHz, 96MHz, 120MHz, 144MHz and 168MHz.

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.

28.6. Function overview

28.6.1. 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.

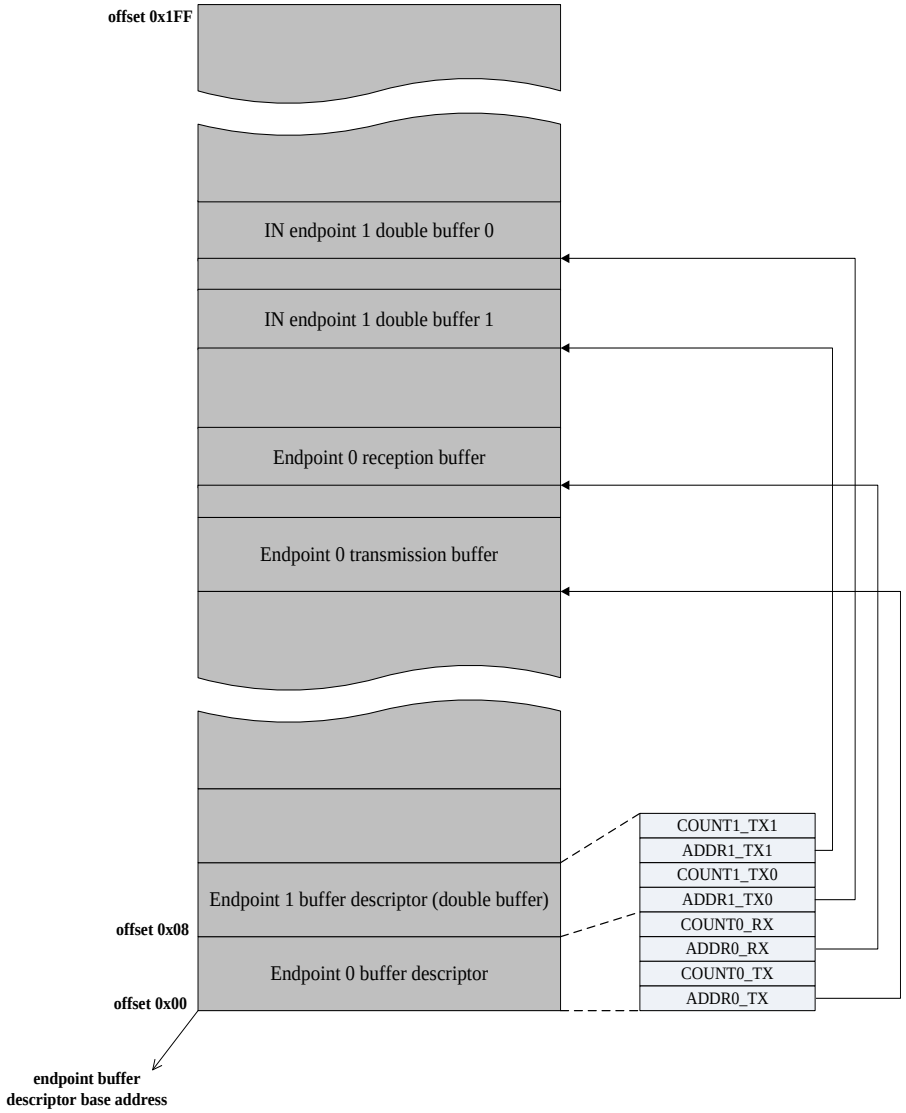
Endpoint buffer descriptor table

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 operation mode. 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 28-2. An example with buffer descriptor table usage (USBD_BADDR = 0)***.

Figure 28-2. An example with buffer descriptor table usage (USBD_BADDR = 0)



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 28-2. Double-buffering buffer flag definition***, the correspondence between USBD_EPxCS register bits and DTG/SW_BUF definition is explained.

Table 28-2. 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	RX_DTG (USBD_EPxCS bit 14)	TX_DTG (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 28-3. Double buffer usage

Endpoint Type	DTG	SW_BUF	Packet buffer used by the USB peripheral	Packet buffer used by the application software
OUT	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.
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.

28.6.2. 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 handle OUT and SETUP tokens in similar way, the difference details about SETUP packets would be shown 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.

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.

According to USB specification, device isn't allowed to abort current command and then start new command, so that device must answer a SETUP packet with an ACK handshake packet, not with a NAK or STALL handshake packet.

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 there is unsuccessfully handling data transfer about previously issued request, the USB discard SETUP token and regard current condition as error, and then urge the host to send the request token again

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. 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.

28.6.3. 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 as 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 USBD_INTF 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 USBD_DADDR 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 consumption 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.

Link Power Management (LPM) level L1

In order to optimize power consumption in SUSPEND/RESUME state, USB 2.0 has achieved Link Power Management (LPM). LPM includes 4 states from L0 to L3. LPM L1 state (sleep state) is the new power management state.

A device will go into the L1 state if the host sends a successful LPM transaction. L1 does not impose any specific power draw requirements on the attached device.

For more details, please refer to USB2_LinkPowerManagement_ECN.

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.

28.6.4. 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_BADDR 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 BLKSIZ and EPRXCNT 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 EPTXCNT 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.

28.7. Registers definition

USBD base address: 0x4000 5C00

28.7.1. 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	L1REQIE	Reserved	L1RSREQ	RSREQ	SETSPS	LOWM	CLOSE	SETRST
rw	rw	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	L1REQIE	LPM L1 state request interrupt enable 0: LPM L1 state request interrupt disabled 1: Interrupt generated when L1REQ bit in USBD_INTF register is set.
6	Reserved	Must be kept at reset value.
5	L1RSREQ	LPM L1 resume request MCU can set this bit to send a LPM L1 resume signal to the host. After the signaling ends, this bit is cleared by hardware.
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.

28.7.2. 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	L1REQ	Reserved		DIR	EPNUM[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r			

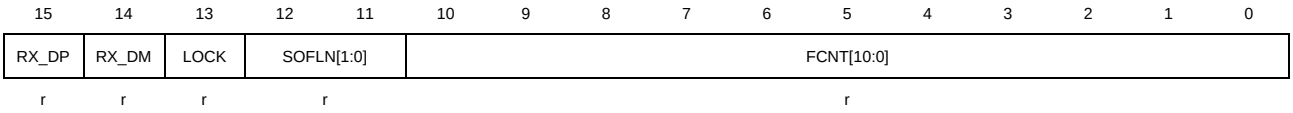
Bits	Fields	Descriptions
15	STIF	Successful transfer interrupt flag This bit set by hardware when a successful transaction completes
14	PMOUIF	Packet memory overrun/underrun interrupt flag This bit set by hardware to indicate that the packet memory is inadequate to hold transfer data. The software writes 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	L1REQ	Set by the hardware when LPM L1 transaction is successfully received and acknowledged. The software writes 0 to clear this bit.
6:5	Reserved	Must be kept at reset value.
4	DIR	Direction of transaction Set by the hardware to indicate the direction of the transaction 0: OUT type 1: IN type
3:0	EPNUM[3:0]	Endpoint Number Set by the hardware to identify the endpoint which the transaction is directed to

28.7.3. 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)



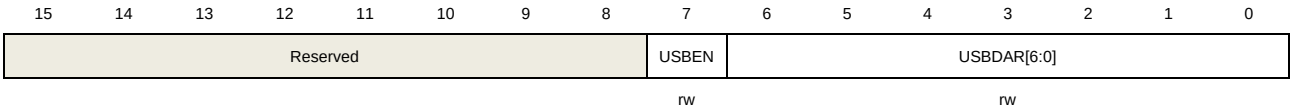
Bits	Fields	Descriptions
15	RX_DP	Receive data + line status Represent the status on the DP line
14	RX_DM	Receive data - line status Represent the status on the DM line
13	LOCK	Locked the USB Set by the hardware indicate that at the least two consecutive SOF have been received
12:11	SOFLN[1:0]	SOF lost 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.

28.7.4. 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)



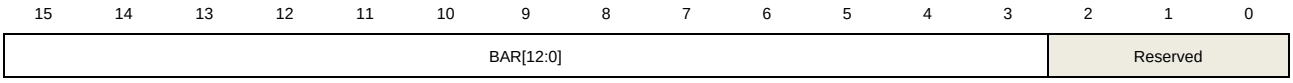
Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7	USBEN	USB device enable Set by software to enable the USB device 0: The USB device disabled. No transactions handled. 1: The USB device enabled.

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

28.7.5. 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)

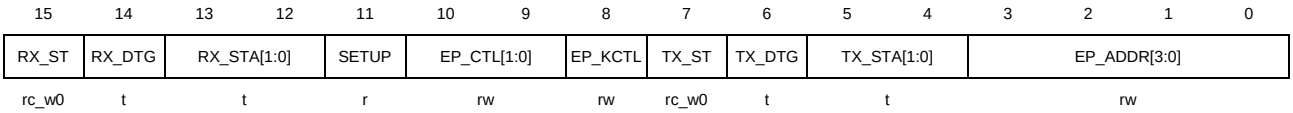


Bits	Fields	Descriptions
15:3	BAR[12:0]	Buffer address Start address of the allocation buffer(512byte on-chip SRAM), used for buffer descriptor table, packet memory
2:0	Reserved	Must be kept at reset value.

28.7.6. 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)



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 28-4. 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 28-5. 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 28-6. Endpoint kind meaning

EP_CTL[1:0]		EP_KCTL Meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT

Table 28-7. 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

28.7.7. USBD endpoint x transmission buffer address register (USBD_EPxTBADDR), x can be in [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

28.7.8. USBD endpoint x transmission buffer byte count register (USBD_EPxTBCNT), x can be in [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

28.7.9. USBD endpoint x reception buffer address register (USBD_EPxRBADDR), x can be in [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

28.7.10. USBD endpoint x reception buffer byte count register (USBD_EPxRBCNT), x can be in [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

28.7.11. USBD LPM control and status register (USBD_LPMCS)

Address offset: 0x54

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												CHIRD			
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RHIRD				BLSTAT[3:0]				REMWK	Reserved	LPMACK	LPMEN
				r				r				r		rw	rw

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:16	CHIRD	Configured HIRD value
15:12	Reserved	Must be kept at reset value
11:8	RHIRD	Received HIRD value
7:4	BLSTAT[3:0]	bLinkState value This filed contain the bLinkState value received with last ACKed LPM token.
3	REMWK	bRemoteWake value This bit contains the bRemoteWake value received with last ACKed LPM token
2	Reserved	Must be kept at reset value.
1	LPMACK	LPM token acknowledge enable 0: the valid LPM token will be NYETed. 1: the valid LPM token will be ACKed. The NYET/ACK will be returned only on a successful LPM transaction: No errors in both the EXT token and the LPM token (else ERROR). A valid bLinkState = 0001B (L1) is received (else STALL)
0	LPMEN	LPM support enable This bit is set by the software to enable the LPM support within the USB device. If this bit is set to 0, no LPM transactions are handled.

29. Universal serial bus High-Speed interface (USBHS)

The USBHS is only available on CL series.

29.1. Overview

USB High-Speed (USBHS) controller provides a USB-connection solution for portable devices. USBHS supports both host and device modes, as well as OTG mode with HNP (Host Negotiation Protocol) and SRP (Session Request Protocol). USBHS contains an embedded USB PHY internal which can be configured as High-Speed or Full-Speed. USBHS supports all the four types of transfer (control, bulk, Interrupt and isochronous) defined in USB 2.0 protocol. There is also a DMA engine operating as an AHB bus master in USBHS to speed up the data transfer between USBHS and system. For Full-Speed operation, battery charging detection (BCD), attach detection protocol (ADP), and link power management (LPM) are also supported.

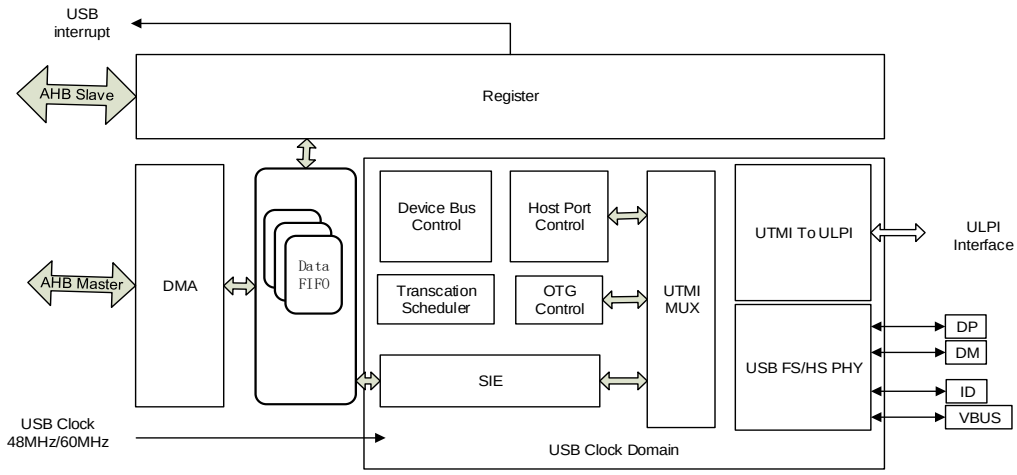
29.2. Characteristics

- Supports USB 2.0 Host mode at High-Speed(480Mb/s), Full-Speed(12Mb/s) or Low-Speed(1.5Mb/s)
- Supports USB 2.0 device mode at High-Speed(480Mb/s) or 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
- Supports high-bandwidth interrupt and isochronous transfers
- Includes USB transaction scheduler in host mode to handle USB transaction request efficiently
- Includes a 4KB FIFO RAM
- Supports 12 channels in host mode
- Contains 2 transmit FIFOs (periodic and non-periodic) and one receive FIFO (shared by all channels) in host mode
- Contains 6 transmit FIFOs (one for each IN endpoint) and one receive FIFO (shared by all OUT endpoints) in device mode
- Supports PING protocol in host mode when operates at High-Speed
- Supports 6 OUT and 6 IN Endpoints in device mode
- Supports remote-wakeup in device mode
- Include a USB PHY with OTG protocol supported
- Include an internal DMA scheduler and engine to perform data copy between USBHS and system per the application’s request
- Time intervals of SOFs is dynamic adjusted in host mode
- Able to output SOF pulse to PAD
- Able to detect ID level and VBUS voltage

- Needs external component to supply power for connected USB device in Host mode or OTG A-device
- Supports charging port detection (BCD) described in Battery Charging Specification Revision 1.2
- Supports Attach Detection Protocol (ADP) described in USB On-The-Go Supplement, Revision 2.0
- Supports Link Power Management (LPM) described in USB 2.0 Link Power Management Addendum and Errata for USB 2.0 ECN: Link Power Management (LPM)

29.3. Block diagram

Figure 29-1. USBHS block diagram



29.4. Signal description

Table 29-1. USBHS signal description

I/0 port	Type	Description	Note
VBUS	Input	Bus power port	For internal PHY only
DM	Input/Output	Differential data line - port	For internal PHY only
DP	Input/Output	Differential data line + port	For internal PHY only
ID	Input	USB identification: Mini connector identification port	For internal PHY only
ULPI_D[7:0]	Input/Output	ULPI Data line	For external ULPI PHY
ULPI_NXT	Input	ULPI next line	For external ULPI PHY
ULPI_DIR	Input	ULPI Direction	For external ULPI PHY
ULPI_STP	Output	ULPI Stop	For external ULPI PHY
ULPI_CLK	Input	ULPI Clock	For external ULPI PHY

29.5. Function overview

29.5.1. USBHS PHY selection, clocks and working modes

USBHS can operate as a host, a device or a DRD (Dual-role-Device) and supports two types of connection: internal embedded PHY and external ULPI PHY. The application choose to use either the internal embedded PHY or the external ULPI PHY according to the demand.

The application may limit the maximum speed of the internal PHY or the external ULPI PHY to Full-Speed using SPDFSLS bit in USBHS_HCTL register in host mode or DS[1:0] in USBHS_DCFG register in device mode.

Table 29-2. USBHS supported speeds

Register configuration		Host supported speed	Device support speed
EMBPHY_FS=1 EMBPHY_HS=0 (Internal FS PHY)		Full-Speed Low-Speed	Full-Speed
EMBPHY_FS=0 EMBPHY_HS=1 (Internal HS PHY)	DS =01 (device mode) SPDFSLS=1(host mode)	Full-Speed Low-Speed	Full-Speed
	DS =00(device mode) SPDFSLS=0(host mode)	High-Speed Full-Speed Low-Speed	High-Speed Full-Speed
EMBPHY_FS=0 EMBPHY_HS=0 (External ULPI PHY)	DS =01 (device mode) SPDFSLS=1(host mode)	Full-Speed Low-Speed	Full-Speed
	DS =00(device mode) SPDFSLS=0(host mode)	High-Speed Full-Speed Low-Speed	High-Speed Full-Speed

The application control the working modes of USBHS: force host，force device by setting FHM and FDM bits in USBHS_GUSBCS register. When both bits are cleared, USBHS works in OTG mode, which is the default mode after system reset.

Internal embedded PHY

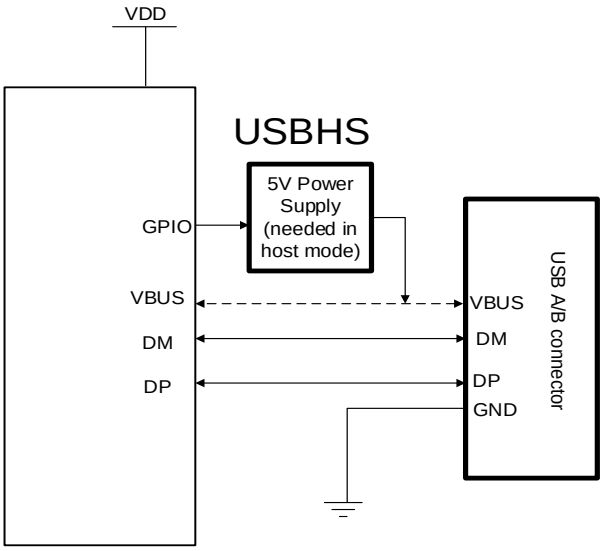
USBHS includes an internal embedded PHY. The internal embedded PHY supports High-Speed, Full-Speed and Low-Speed in host mode, and High-Speed, Full-Speed in device mode, supports OTG protocol with HNP and SRP. Software needs to set EMBPHY_FS bit and reset EMBPHY_HS in USBHS_GUSBCS register to use this PHY in FS mode, or reset EMBPHY_FS bit and set EMBPHY_HS to use this PHY in HS mode. If internal embeddedPHY is selected, the USB clock used for the USBHS needs to be 48MHz in FS mode and 60MHz in HS mode. The 48MHz USB clock is generated from internal clocks in system, and its source and divider factors are configurable in RCU, while the 60MHz USB clocks is generated from 480MHz PLLUSB.

Note: In the course of configuring 480MHz clock, it is not recommend to select CK_IRC48M

in PLLUSBPRESEL bit of RCU_ADDCFG register.

The pull-up and pull-down resistors are already integrated into the internal PHY and controlled by USBHS automatically based on the current mode (host, device or OTG mode) and connection status. A typical connection using internal PHY is shown in ***Figure 29-2. Connection using internal embedded PHY with host or device mode***.

Figure 29-2. Connection using internal embedded PHY with host or device mode

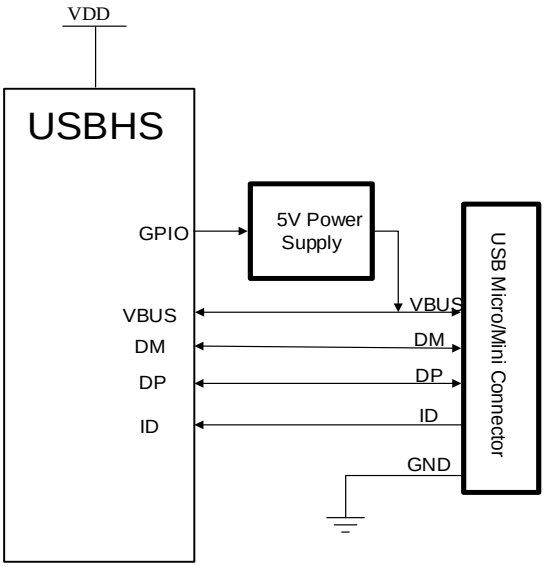


When USBHS works in host mode (FHM bit is set and FDM bit is cleared), the VBUS is 5V power pin defined in USB protocol. The internal PHY cannot supply 5V VBUS power and only has some voltage comparers, charge and dis-charge circuit on VBUS line. If application needs to supply USB power, an external power supply IC is needed. The VBUS connection between USBHS and the USB connector can be omitted in host mode because USBHS doesn't detect the voltage level on VBUS pin and always assumes that the 5V power is present.

When USBHS works in device mode (FHM bit is cleared and FDM bit is set), the VBUS detection circuit is decided by VDEN bit in USBHS_GCCFG register. If the device does not need to detect the voltage on VBUS pin, it may set the VDEN bit and free the VBUS pin for other use. Otherwise, the VBUS connection cannot be omitted, and USBHS continuously monitor the VBUS voltage and will immediately switch off the pull-up resistor on DP line once the VBUS voltage falls below the needed valid value. This will cause a disconnection.

The OTG mode connection is described in the ***Figure 29-3. Connection using internal embedded PHY with OTG mode***. When USBHS works in OTG mode, the FHM, FDM bits in USBHS_GUSBCS and VDEN bit in USBHS_GCCFG should be cleared. In this mode, the USBHS needs all the four pins: DM, DP, VBUS and ID, and uses several voltage comparers to monitor the voltage on these pins. USBHS also includes VBUS charge and discharge circuit to perform SRP request described in OTG protocol. The OTG A-Device or B-Device is decided by the level of ID pins. USBHS controls the pull-up or pull-down resistor during the HNP protocol.

Figure 29-3. Connection using internal embedded PHY with OTG mode

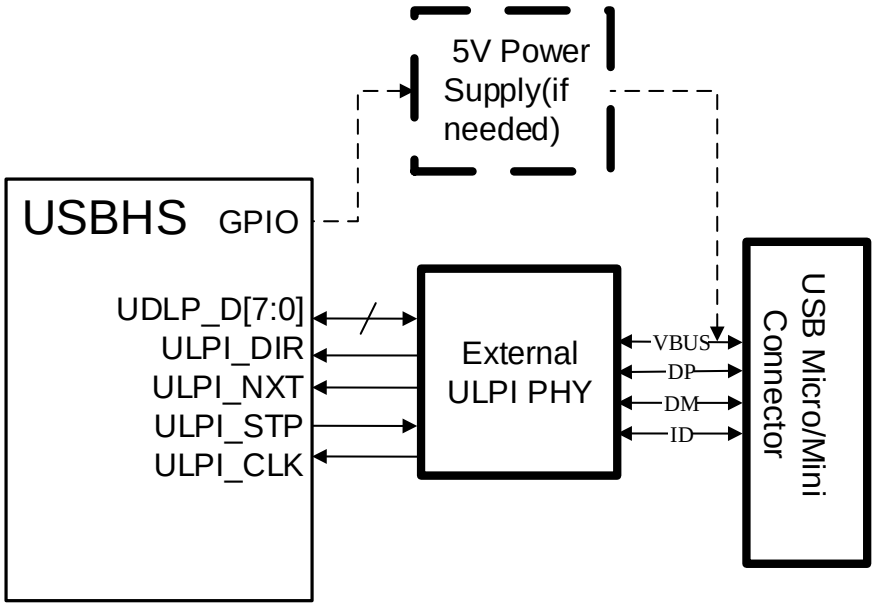


External ULPI PHY

USBHS provides a ULPI interface for external PHY integration. An external High-Speed ULPI PHY is needed to support High-Speed USB applications. With external ULPI PHY, USBHS supports High-Speed host and device, all the modes described in internal embedded PHY.

Software needs to clear the EMBPHY_FS bit and EMBPHY_HS bit in USBHS_GUSBCS register to enable the ULPI interface. When ULPI mode enabled, the USB clock which introduced from the ULPI_CLK pin needs to be 60MHz.Software can switch on or off the 60MHz ULPI clock in RCU.

Figure 29-4. Connection using external ULPI PHY

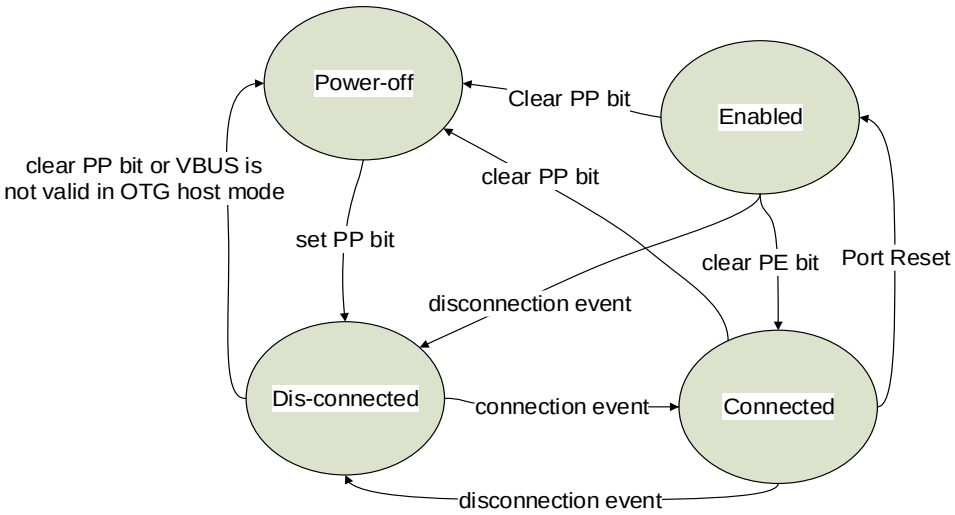


29.5.2. USB host function

USB Host Port State

Host application may control state of the USB port via USBHS_HPCS register. After system initialization the USB port, keep it at power-off state. After PP bit is set by software, the USB PHY (either internal or external) 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 29-5. State transition diagram of host port



Connection, Reset and Speed identification

As a USB host, USBHS will trigger a connection flag for application after a connection is detected and will trigger 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 connection or enabled state.

The USBHS performs speed identification during connection and reset, and the speed information is reported in PS[1:0] bits in USBHS_HPCS register.

If the maximum supported speed is configured to Full-Speed (SPDFSLS = 1), USBHS only performs speed-identification during device connection process and it identifies the device speed from the voltage level of DM or DP. As is described in USB protocol, Full-Speed device pulls up DP line while Low-Speed device pulls up DM line.

If the maximum supported speed is configured to High-Speed (SPDFSLS = 0), USBHS first performs speed-identification during connection. If a Full-Speed connection is detected, the USBHS will try to perform High-Speed identification (CHIRP sequence described in USB 2.0 protocol) during each USB reset sequence after the connection event. So the application on host should perform a USB reset after a connection event and check the PS[1:0] bits again if

it desires to support High-Speed device.

Suspend and resume

USBHS supports suspend state and resume operation. When USBHS port is at enabled state, writing 1 to PSP bit in USBHS_HPCS register will cause USBHS to enter suspend state. In suspend state, USBHS 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 USBHS_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 USBHS_GINTF and the USBHS wake up interrupt will be triggered if a host in suspend state detects a remote wakeup signal.

SOF generate

USBHS 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 for Full-Speed links, and every 125 μs for High-Speed links.

Each time after USBHS enters into enabled state, it will send SOF packet using the time defined by USB 2.0 protocol. While, application may adjust the length of a frame or a micro-frame by writing to FRI[15:0] in USBHS_HFT registers. The FRI bits define the number of USB clock cycles in a frame or micro-frame and application should calculate the value based on the frequency of USB clock used by USBHS. The FRT[14:0] bits reflect the remaining clock cycles of the current frame or micro-frame and stops to change during suspend state.

USBHS is able to generate a pulse signal each SOF packet and output it to a pin. The pulse length is 12 HCLK cycle. If application desires to use this function, it needs to set SOFOEN bit in USBHS_GCCFG register and configure the related pin registers in GPIO.

USB Channels and Transactions

USBHS includes 12 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 is configured in channel related registers such as USBHS_HCHxCTL and USBHS_HCHxLEN.

USBHS supports all the four kinds of transfer types: control, bulk, interrupt 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, USBHS includes two request queues: periodic request queue and non-periodic request queue, in order 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.

In non-DMA mode, application needs to write packet into data FIFO via AHB register interface if it wants to start an OUT transaction on USB bus. USBHS hardware will automatically generate a transaction request entry in request queue after the application writes a whole packet. In DMA mode, application only needs to configure the channel property and channel data buffer address, and the DMA engine in USBHS performs the packet data copy and

request entry generation. USBHS automatically generate IN request entries when the application enable an IN channel.

The request entries in request queue are processed in order by transaction control module. USBHS always try to process periodic request queue first, then process non-periodic request queue.

After a start of frame USBHS 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 ensure the bandwidth of periodic transactions in a frame or micro-frame. Each time the USBHS reads and pop a request entry from request queue. If this is a channel disable request, it immediately disables the channel and prepare to process next entry.

If the current request is a transaction request and the USB bus time is enough for this transaction, USBHS will employ SIE to generate this transaction on USB bus.

When the required bus time by the current request is not enough in the current frame, if this is a periodic request, USBHS stops the processing of periodic queue and starts to process non-periodic request. If this is a non-periodic queue the USBHS will stop to process any queue and wait until the end of current frame.

LPM

This addendum for USB defines power management states (LPM states) and mechanisms to affect state changes that are used by hosts and hubs to efficiently manage bus and system power. LPM simply adds a new feature and bus state sleep state (L1) that co-exists with the USB2.0 defined suspend (L2)/resume.

L1 is similar to L2 but supports finer granularity in use. Entry to L1 is started by a request to a hub or host port to transition to L1. A LPM transaction is sent to the downstream device. The requested transition can only occur if the device response with an ACK handshake. Exit from L1 is via remote wake, resume signaling, reset signaling or disconnect. Either the host or device can initiate resume signaling when in L1. Although the signaling levels of resume are the same as L2, the duration of the signaling and transitional latencies associated with the L1 to L0 (active state) transition are much shorter.

29.5.3. USB device function

USB Device Connection

In device mode USBHS stays at power-off state after initialization. After connected to a USB host with 5V power supply present on VBUS pin or setting VDEN bit in USBHS_GCCFG register, USBHS enters into powered state. USBHS begins to switch on the pull-up resistor on DP line, host side will detect a connection event.

Reset and Speed-Identification

The USB host always starts a USB reset after it detects a device connection, USBHS in device

mode will trigger a reset interrupt for software after it detects the reset event on USB bus.

If the maximum supported speed is configured to Full-Speed (DS[1:0] = 01 in USBHS_DCFG register), USBHS will operate as a Full-Speed device. If the maximum supported speed is configured to High-Speed (DS[1:0] = 00 in USBHS_DCFG register), USBHS device tries to start a speed-identification (a chirp handshake described in USB 2.0 protocol) with host during reset sequence. If the chirp handshake with host successes, the device enters High-Speed mode, otherwise, remains at Full-Speed mode.

After reset sequence, speed-identification process completes, USBHS triggers an ENUMF interrupt in USBHS_GINTF register and reports current enumerated device speed in ES bits in USBHS_DSTAT register. If software want to implement a High-Speed device, it must wait ENUMF interrupt first, then read the ES[1:0] bits to get the speed-identification result.

As required by USB 2.0 protocol, USBHS doesn't support Low-Speed in device mode.

Suspend and Wake-up

A USB device will enter into suspend state after the USB bus stays at IDLE state and has no change on data lines for 3ms. When USB device is in suspend state, software can switch off most of its clock to save power. The USB host is able to wake up the suspended device by generating a resume signal on USB bus. USBHS is able to detect the resume signal and triggers the WKUPIF flag in USBHS_GINTF register and the USBHS wake up interrupt.

In suspend mode, USBHS is also able to remote wake-up the USB bus. Software may set RWKUP bit in USBHS_DCTL register to sends 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

USBHS supports soft disconnection. After the device is power on, USBHS will switch on the pull-up resistor on DP line and this will cause the host to detect the connection. Then, software is able to force a disconnection by setting the SD bit in USBHS_DCTL register. After the SD bit is set, if the current device speed is High-Speed, USBHS will first return back to Full-Speed device and then switch off the pull-up resistor on DP line, and if current speed is Full-Speed, USBHS will directly switch off the pull-up resistor. This will cause USB host to detect a disconnection on USB bus.

SOF tracking

When USBHS receives a SOF packet from USB bus, it triggers a SOF interrupt and begins to count the bus time by using local USB clock. The frame number of the current frame is reported in FNRSOF[13:0] in USBHS_DSTAT register. When the USB bus time reaches EOF1 or EOF2 point (End of Frame, described in USB 2.0 protocol), USBHS will trigger a interrupt EOPFIF in USBHS_GINTF register. Software is able to use these flags and registers to get current bus time and position information.

BCD

Charging port detection (BCD) described in Battery Charging Specification Revision 1.2 is

supported. In order for PD (portable device) to determine how much current it is allowed to draw from an upstream USB port, there need to be mechanisms that allow the PD to distinguish between a Standard Downstream Port and a Charging Port.

In BCD mechanisms, USB VBUS detection (VD), data contact detection (DCD), primary detection (PD) and secondary detection (SD) are included. The control and configuration bits about BCD is reported in USBHS_GCCFG register.

29.5.4. OTG function overview

USBHS supports OTG function described in OTG protocol 1.3/2.0. 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 for 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 in a Standard-A plug. The B-Device is a peripheral at the start of a session. USBHS uses the voltage level of ID pin to judge A-Device or B-Device. The ID status is reported in IDPS bit in USBHS_GOTGCS register. For the details of states transfer between A-Device and B-Device, please refer to OTG 1.3/2.0 protocol.

HNP

The Host Negotiation Protocol (HNP) allows the host function to be transferred between two directly connected On-The-Go devices and eliminates the need for a user to switch the cable connections in order to allow a change in control of communications between the devices. HNP will typically be initiated 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 Host or Peripheral, depending upon which type of plug (Micro-A plug for Host, Micro-B plug for Peripheral) is inserted. By utilizing the Host Negotiation Protocol (HNP), an On-The-Go B-Device, which is the default Peripheral, may make a request to be Host. The process for this exchange of the role of Host is described in this section. This protocol eliminates the need for the user to swap the cable connection in order to change the roles of the connected devices.

When USBHS is in OTG A-Device host mode and it wants to give up its host role, it may first set PSP bit in USBHS_HPCS register to make the USB bus enter suspend status. Then, the B-device will enter suspend state after 3ms. If the B-Device wants to changes to host, software needs to set HNPREQ bit in USBHS_GOTGCS register and the USBHS will begin to perform HNP protocol on bus, and at last, the result of HNP is reported in HNPS bit in USBHS_GOTGCS register. Besides, software is always able to get the current role (host or peripheral) from COPM bit in USBHS_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 is reported in ASV and BSV bits in USBHS_GOTGCS register.

Software may set SRPREQ bit in USBHS_GOTGCS register to start a SRP request when USBHS is in B-Device OTG mode and USBHS will generate a success flag SRPS in USBHS_GOTGCS register if the SRP request successes.

When USBHS is in OTG A-Device mode and it detects an SRP request from a B-Device, it sets a SESIF flag in USBHS_GINTF register. The software should prepare to switch on the 5V power supply for VBUS pin after it gets this flag.

ADP

Attach Detection Protocol (ADP) is a protocol that allows a local device to detect when a remote device has been attached or detached. The remote device can be any USB device. ADP operates by detecting the change in VBUS capacitance that occurs when two devices are attached or detached. The capacitance is detected by first discharging the VBUS line, and then measuring the time it takes VBUS to charge to a known voltage with a known current source. A change in capacitance is detected by looking for a change in the charge time.

Software may set bit ADPMEN, ADPEN and ENAPRB to perform ADP probe, and should perform at least one ADP probe cycle in order to obtain an initial value for TADP_RISE when an ADP-capable A-device or B-device is first powered up. For B-device, ADP sense can be performed by setting bit ENASNS. If RITM in USBHS_ADPCTL register changes, it shows that a remote device has been attached or detached.

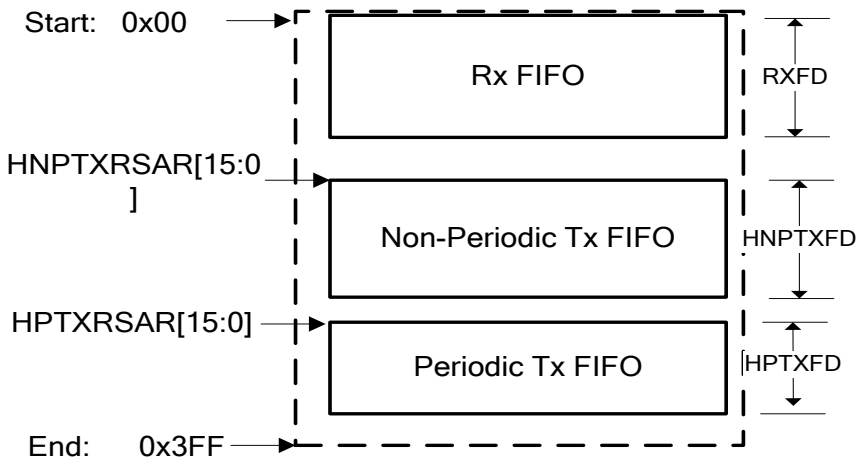
29.5.5. Data FIFO

The USBHS include a 4K bytes data FIFO to store packet data. The data FIFO is implemented by using an internal SRAM.

Host Mode

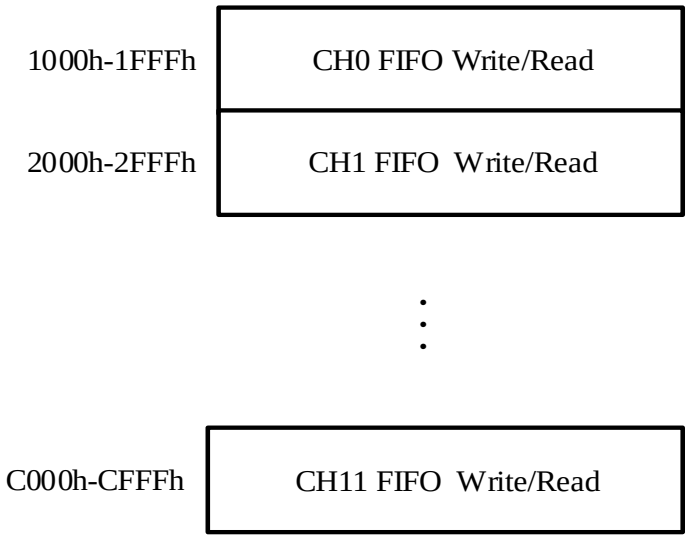
In host mode the data FIFO space is divided into 3 parts: Rx FIFO for received packet, Non-Periodic Tx FIFO for non-period transmission packet and Periodic Tx FIFO for periodic transmission packet. All IN channels shares the Rx FIFO for receiving packets. All the periodic OUT channels share the periodic Tx FIFO to transmit packets. All the non-periodic OUT channels share the non-Periodic FIFO for transmit packets. Software should configure the size and start offset of these data FIFOs by use these registers: USBHS_GRFLEN, USBHS_HNPTFLEN and USBHS_HPTFLEN. ***Figure 29-6. 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 29-6. Host mode FIFO space in SRAM



In DMA mode, DMA engine is responsible for packet data copy between system memory and the internal data FIFOs. In non-DMA mode the application needs to manually write packet data into or read packet from the data FIFOs. USBHS provides a special register area for software to write and read the internal data FIFO. ***Figure 29-7. Host mode FIFO access register map*** describes the register memory area for data FIFO access. The addresses in the figure are in term of byte. Each channel has its own FIFO access register space, although all Non-periodic channels share the same FIFO and all the Periodic channels share the same FIFO. This is important for USBHS to know the current pushed packet belongs to which channel. Rx FIFO is also able to be accessed by using USBHS_GRSTATR/USBHS_GRSTATP register.

Figure 29-7. Host mode FIFO access register map

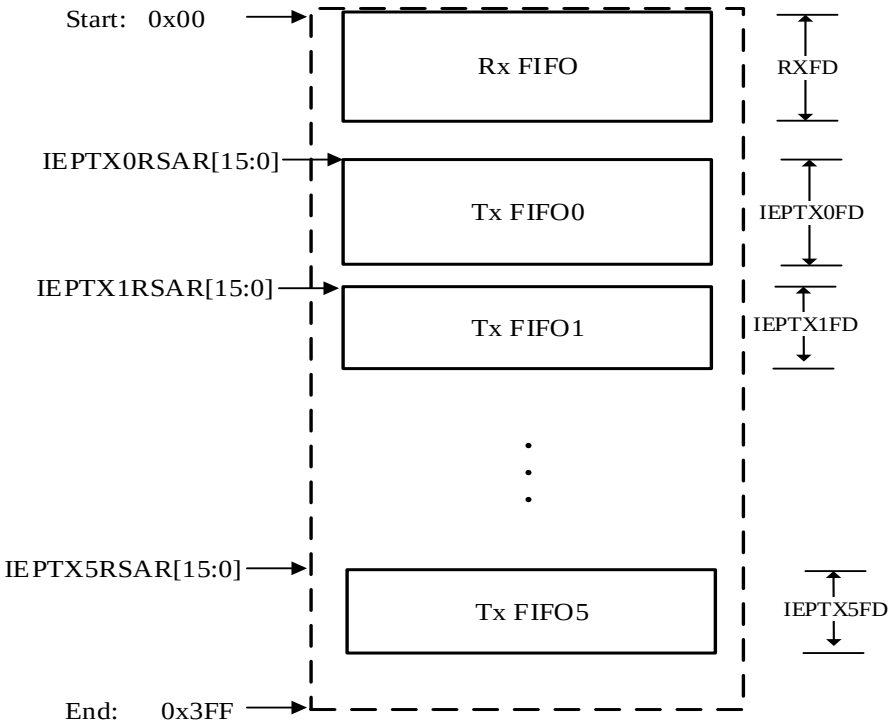


Device mode

In device mode, the data FIFO is divided into several parts: one Rx FIFO, and 6 Tx FIFOs (one for each IN endpoint). All the OUT endpoints share the Rx FIFO for receiving packets. Software should configure the size and start offset of these data FIFOs by using

USBHS_GRFLEN and USBHS_DIEPxTFLEN (x=0...5) registers. ***Figure 29-8. 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 29-8. Device mode FIFO space in SRAM



In DMA mode, DMA engine is responsible for packet data copy between system memory and the internal data FIFOs. In non-DMA mode the application needs to manually write packet data into or read packet from the data FIFOs. USBHS provides a special register area for software to write and read the internal data FIFO. ***Figure 29-9. Device mode FIFO access register map*** describes the register memory area for data FIFO access. The addresses in the figure are in term of byte. Each endpoint has its own FIFO access register space. Rx FIFO is also able to be accessed by using USBHS_GRSTATR/USBHS_GRSTATP register.

Figure 29-9. Device mode FIFO access register map

29.5.6. DMA function

This section describes the DMA scheduler and DMA engine in USBHS.

DMA Requests and Scheduler

DMA function is enabled by setting DMAEN bit in USBHS_GAHBCS register. When an IN/OUT channel or IN endpoint is properly configured and enabled, or the Rx FIFO is not empty, USBHS will generate DMA request. There is a DMA scheduler in USBHS responsible for responding to theses DMA requests.

There may be several requests simultaneously and the DMA scheduler arbitrates among these requests. These requests are sorted into 3 kinds: Rx FIFO DMA request, periodic transfer DMA requests and non-periodic transfer DMA requests. Rx FIFO DMA request takes the highest priority, and periodic transfer DMA requests take the medium priority, and non-periodic transfer DMA requests take the lowest priority when arbitration. DMA scheduler performs round-robin arbitration method within the periodic or non-periodic transfer DMA requests.

As is described above, DMA will automatically handle the Rx FIFO not empty event, so software should ignore the RXFNEIF flag in USBHS_GINTF register in DMA mode.

DMA Engine

Receive:

In host or device mode, once Rx FIFO DMA request gets arbitration, DMA engine begins to read a packet or a status entry from Rx FIFO. For data packet, DMA write the data into the specified system address configured in the HCHxDMAADDR register or DIEPxDMAADDR/DOEPxDMAADDR register. For status entry, DMA will generate the specified flags or interrupts on related channels or endpoints.

Host Transfer:

When a periodic or non-periodic IN channel DMA request gets arbitration, DMA writes IN request entries into the periodic or non-periodic request queue. After the desired IN transfers completes, or an AHB/USB bus error occurs, DMA halts the specified channel and generate TF and CH flags in USBHS_HCHxINTF register. The received packet during IN transfers copied into system memory after the Rx FIFO DMA request is generated, as described above.

When an OUT periodic or non-periodic channel DMA request gets arbitration, DMA reads packet data from system memory and writes to internal Tx FIFO. DMA always writes an OUT request entry into the request queue when it finishes a packet data copying. After the desired OUT transfers completes, or an AHB/USB bus error occurs, DMA halt the specified channel and generate TF and CH flags in USBHS_HCHxINTF register.

Device Transfer:

In device mode, when an IN endpoint DMA request gets arbitration, DMA reads packet data from system memory and writes to the endpoint's Tx FIFO. When USBHS gets an IN token on an IN endpoint, it transmits the packet copied by DMA engine.

29.5.7. Operation guide

This section describes the advised operation guide for USBHS.

Host mode

Global register initialization sequence

1. Program USBHS_GAHBCS register according to application's demand, such as: whether to enable DMA, burst type of DMA transfer and the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.
2. Program USBHS_GUSBCS register according to application's demand, such as: the operation mode (host, device or OTG) and some parameters of OTG，ULPI and USB protocols.
3. Program USBHS_GCCFG register according to application's demand.
4. Program USBHS_GRFLEN, USBHS_HNPTFLEN_DIEP0TFLEN and USBHS_HPTFLEN registers to configure the data FIFOs according to application's demand.
5. Program USBHS_GINTEN register to enable Mode Fault and Host Port interrupt and set GINTEN bit in USBHS_GAHBCS register to enable global interrupt.
6. Program SPDFSLS bit in USBHS_HCTL register to select whether to limit the device speed to Full-Speed.
7. Program USBHS_HPCS register and set PP bit.

8. Wait for a device's connection, and once a device is connected, the connection interrupt PCD in USBHS_HPCS register will be triggered. Then set PRST bit to perform a port reset. Wait for at least 10ms and then clear PRST bit.

9. Wait PEDC interrupt in USBHS_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 USBHS_HFT register if software want to change the SOF interval.

Channel initialization and enable sequence

1. Program USBHS_HCHxCTL register with desired transfer type, direction, packet size, etc. Ensure that CEN and CDIS bits keep cleared during configuration.

2. Program USBHS_HCHxINTEN register. Set the desired interrupt enable bits.

3. If DMA is enabled, program USBHS_HCHxDMAADDR register.

4. Program USBHS_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 USBHS_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, software may program TLEN as a maximum possible value supported by Rx FIFO.

5. Set CEN bit in USBHS_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. USBHS will generate a channel disable request entry in request queue after the register setting operation. When the request entry reaches to the top of request queue, it is processed by USBHS immediately:

For OUT channel, the specified channel will be disabled immediately. Then, a CH flag will be generated and the CEN and CDIS bits will be cleared by USBHS.

For IN channels, USBHS 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 with DMA disabled

1. Initialize USBHS global registers.

2. Initialize the channel.

3. Enable the channel.
4. After the IN channel is enabled by software, USBHS generates a Rx request entry in the corresponding request queue.
5. When the Rx request entry reaches to the top of the request queue, USBHS begins to process this request entry. If bus time for the IN transaction indicated by the request entry is enough, USBHS starts the IN transaction on USB bus.
6. If the IN transaction finishes successfully (ACK handshake received), USBHS pushes the received data packet into the Rx FIFO and triggers ACK flag. Otherwise, the status flag (NAK) report the transaction result.
7. If the IN transaction described in step 5 is successful and PCNT is larger than 1 in step2, software should return to step 3 and continues to receive the remaining packets. If the IN transaction described in step 5 is not successful, software should return to step 3 to re-receive the packet again.
8. After all the transactions in a transfer are successful received on USB bus, USBHS push a TF status entry into the Rx FIFO on top of the last packet data. After software reads and pops all the received data packet, and at last, the TF status entry, USBHS 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.

IN transfers operation sequence with DMA enabled

1. Initialize USBHS global registers.
2. Initialize and enable the channel.
3. After the IN channel is enabled by software, USBHS begins to generate Rx request entry in the corresponding request queue.
4. USBHS processes the request entries in request queue one by one and perform the indicated IN transactions on USB bus.
5. When a IN transaction gets a NAK handshake, the DMA is able to re-send IN tokens automatically until that USBHS get the desired number of packets .
6. After USBHS gets the desired number of packets specified by PCNT in USBHS_HCHxLEN register, USBHS generates TF and CH flags to indicate that the transfer successfully finishes and the channel is disabled. If USB bus error or DMA write error occurs during these transactions, DMA will trigger related error flags, stops the processing for this channel, disable this channel and at last, trigger the CH flag.

Note: In DMA mode, software should not enable or process the RXFNEIF interrupt because the DMA will automatically process the Rx FIFO.

OUT transfers operation sequence with DMA disabled

1. Initialize USBHS 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, USBHS generates a Tx request entry in the corresponding request queue and decrease the TLEN field in USBHS_HCHxLEN register with the written packet's size.
4. When the request entry reaches to the top of the request queue, USBHS begins to process this request entry. If bus time for the transaction indicated by the request entry is enough, USBHS starts the OUT transaction on USB bus.
5. When the OUT transaction indicated by the request entry finishes on USB bus, PCNT in USBHS_HCHxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flag (NAK) report the transaction result.
6. If the OUT transaction described in step 5 is successful and PCNT is larger than 1 in step2, software should return to step 3 and continues to send the remaining packets. If the OUT transaction described in step 5 is not successful, software should return to step 3 to resend the packet again.
7. After all the transactions in a transfer are successful sent on USB bus, USBHS 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.

OUT transfers operation sequence with DMA enabled

1. Initialize USBHS global registers.
2. Initialize and enable the channel.
3. DMA in USBHS begins to fetch packets from the address specified by DMAADDR in USBHS_HCHxDMAADDR register and write them into the channel's Tx FIFO (Periodic Tx FIFO or non-periodic Tx FIFO). Each time a whole packet data is written into the FIFO, USBHS generates a Tx request entry in the corresponding request queue and decrease the TLEN field in USBHS_HCHxLEN register with the written packet's size.
4. USBHS processes the request entries in request queue one by one and sends out the indicated transactions on USB bus.
5. When a transaction gets a NAK or NYET handshake, the DMA is able to re-fetch and re-send the packet as well as perform PING protocol automatically.
6. If all the transactions are successful sent on USB bus, USBHS generates TF and CH flags to indicate that the transfer successfully finishes and the channel is disabled. If USB bus error or DMA fetch error occurs during these transactions, DMA will trigger related error flags, stops the processing for this channel, disable this channel and at last, trigger the CH flag.

Note: In DMA mode, software should not enable or process the RXFNEIF interrupt because

the DMA will automatically process the Rx FIFO.

Device mode

Global register initialization sequence

1. Program USBHS_GAHBCS register according to application's demand, such as: whether to enable DMA, burst type of DMA transfer and the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.
2. Program USBHS_GUSBCS register according to application's demand, such as: the operation mode (host, device or OTG) and some parameters of OTG，ULPI and USB protocols.
3. Program USBHS_GCCFG register according to application's demand.
4. Program USBHS_GRFLEN, USBHS_HNPTFLEN_DIEP0TFLEN and USBHS_DIEPxTFLEN registers to configure the data FIFOs according to application's demand.
5. Program USBHS_GINTEN register to enable Mode Fault, Suspend, SOF, Enumeration Done and USB Reset interrupt and then, set GINTEN bit in USBHS_GAHBCS register to enable global interrupt.
6. Program USBHS_DCFG register according to application's demand, such as the device speed and 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 USBHS_GINTF register.
8. Wait for ENUMF interrupt in USBHS_GINTF register and then read ES[1:0] bits in USBHS_DSTAT register to get the current enumerated device speed.

Endpoint initialization and enable sequence

1. Program USBHS_DIEPxCTL or USBHS_DOEPxCTL register with desired transfer type, packet size, etc.
2. Program USBHS_DIEPINTEN or USBHS_DOEPINTEN register. Set the desired interrupt enable bits.
3. If DMA is enabled, program USBHS_DIEPxDMAADDR or USBHS_DOEPxDMAADDR register.
4. Program USBHS_DIEPxLEN or USBHS_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 USBHS_DIEPxCTL 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 OUT endpoint：Because the application doesn't know the actual received data size before the OUT transaction finishes, software may program TLEN as a maximum possible value supported by Rx FIFO.

5. Set EPEN bit in USBHS_DIEPxCTL or USBHS_DOEPxCTL register to enable the endpoint.

Endpoint disable sequence

Software can disable the endpoint anytime when clearing the EPEN bit in USBHS_DIEPxCTL or USBHS_DOEPxCTL register.

IN transfers operation sequence with DMA disabled

1. Initialize USBHS 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, USBHS decreases the TLEN field in USBHS_DIEPxLEN register with the written packet's size.
4. When an IN token is received, USBHS transmit the data packet, and after the transaction finishes on USB bus, PCNT in USBHS_DIEPxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flags report the transaction result.
5. After all the data packets in a transfer are successful sent on USB bus, USBHS generates TF flag to indicate that the transfer successfully finishes and disable the IN endpoint.

IN transfers operation sequence with DMA enabled

1. Initialize USBHS global registers.
2. Initialize and enable the IN endpoint.
3. DMA in USBHS begins to fetch packets from the address specified by DMAADDR in USBHS_DIEPxDMAADDR register and write them into the IN endpoint's Tx FIFO. Each time a whole packet data is written into the FIFO, USBHS decreases the TLEN field in USBHS_DIEPxLEN register with the written packet's size.
4. When an IN token is received, USBHS transmit the data packet, and after the transaction finishes on USB bus, PCNT in USBHS_DIEPxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flags report the transaction result.
5. If all the transactions are successful sent on USB bus, USBHS generates TF and EPDIS flags to indicate that the transfer successfully finishes and the endpoint is disabled. If USB bus error or DMA fetch error occurs during these transactions, DMA will trigger related

error flags.

Note: In DMA mode, software should not enable or process the RXFNEIF interrupt because the DMA will automatically process the Rx FIFO.

OUT transfers operation sequence with DMA disabled

1. Initialize USBHS global registers.
2. Initialize the endpoint and enable the endpoint.
3. When an OUT token is received, USBHS receive the data packet or response with an NAK handshake based on the status of Rx FIFO and register configuration. If the transaction finishes successfully (USBHS receives and saves the data packet into Rx FIFO successfully and sends ACK handshake on USB bus), PCNT in USBHS_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 successful received on USB bus, USBHS push a TF status entry into the Rx FIFO on top of the last packet data. After software reads and pops all the received data packet, and at last, the TF status entry, USBHS generates TF flag to indicate that the transfer successfully finishes and disable the OUT endpoint.

OUT transfers operation sequence with DMA enabled

1. Initialize USBHS global registers.
2. Initialize and enable the OUT endpoint.
3. When an OUT token received, USBHS receive the data packet or response with an NAK handshake based on the status of Rx FIFO and register configuration. If the transaction finishes successfully (USBHS receives and saves the data packet into Rx FIFO successfully and sends ACK handshake on USB bus), PCNT in USBHS_DOEPxLEN register is decreased by 1 and the ACK flag is triggered, otherwise, the status flags report the transaction result.
4. If all the transactions are successful received on USB bus, USBHS generates TF and EPDIS flags to indicate that the transfer successfully finishes and the endpoint is disabled. If USB bus error or DMA write error occurs during these transactions, DMA will trigger related error flags.

Note: In DMA mode, software should not enable or process the RXFNEIF interrupt because the DMA will automatically process the Rx FIFO.

29.6. Interrupts

USBHS has four interrupts: global interrupt, wake-up interrupt, endpoint1 IN interrupt and endpoint1 OUT interrupt.

Global interrupt is the main interrupt software should process, the source flags of the global

interrupt are readable in USBHS_GINTF register and listed in the following ***Table 29-3. USBHS global interrupt***.

Table 29-3. USBHS global interrupt

Interrupt Flag	Description	Operation Mode
SESIF	Session interrupt	Host or device mode
DISCIF	Disconnect interrupt flag	Host Mode
IDPSC	ID pin status change	Host or device mode
LPMIF	LPM interrupt flag	Host or device 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 is able to be triggered when USBHS is in suspend state, even when the USBHS's clocks are stopped. The source of the wake up interrupt is WKUPIF bit in USBHS_GINTF register.

Endpoint 1 IN/OUT interrupts are two special interrupts for endpoint 1. Application can use these two interrupts to make a quick response to the events on endpoint 1. The two interrupts are individually enabled by USBHS_DEP1INT register. And the source of these two interrupts also come from USBHS_DIEP1INTF and USBHS_DOEP1INTF registers, but the enable bits for these flags to generate Endpoint 1 IN/OUT interrupts are in USBHS_DIEP1INTEN and

USBHS_DOEP1INTEN registers.

29.7. Register definition

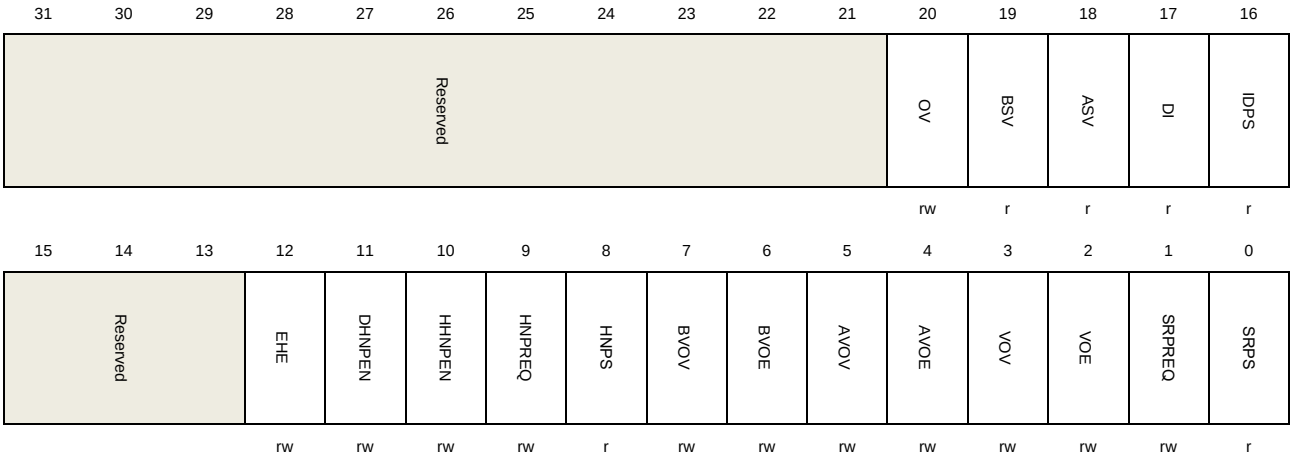
USBHS base address: 0x5000 0000

29.7.1. USBHS global registers

Global OTG control and status register (USBHS_GOTGCS)

Address offset: 0x0000
Reset value: 0x0000 0800

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



Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20	OV	Select OTG version 0: Version 1.3 is selected. Data line pulsing and VBUS pulsing are supported in SRP 1: Version 2.0 is selected. Only data line pulsing is supported in SRP
19	BSV	B-Session Valid (described in OTG protocol). 0: Vbus voltage level of a OTG B-device is below VBSESSVLD 1: Vbus voltage level of a OTG B-Device is not below VBSESSVLD **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 VASESSVLD 1: Vbus voltage level of a OTG A-device is not below VASESSVLD The A-device is as host default 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 occur 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: USBHS is in A-device mode 1: USBHS is in B-device mode **Note:** Accessible in both device and host modes.
15:13	Reserved	Must be kept at reset value
12	EHE	Embedded host enable 0: OTG A-device state is selected 1: Embedded host state is selected
11	DHNPEN	Device HNP enable Enable the HNP function of a B-device. If this bit is cleared, USBHS doesn't start HNP protocol when application set HNPREQ bit in USBHS_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, USBHS 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. Software can clear this bit when HNPEND bit in USBHS_GOTGINTF register is set, by writing zero to it, or clearing the HNPEND bit in USBHS_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 successes and cleared when HNPREQ bit is set. 0: HNP fails 1: HNP successes **Note:** Only accessible in device mode.

7	BVOV	Override value of B-peripheral session valid 0: B-peripheral session valid value is 0 when BVOE = 1 1: B-peripheral session valid value is 1 when BVOE = 1 **Note:** Only accessible in device mode.
6	BVOE	Override enable of B-peripheral session valid 0: Override is disable. Internally B-peripheral session valid received from PHY is selected 1: Override is enable. Internally B-peripheral session valid received from PHY is overridden with BVOV **Note:** Only accessible in device mode.
5	AVOV	Override value of A-peripheral session valid 0: A-peripheral session valid value is 0 when AVOE = 1 1: A-peripheral session valid value is 1 when AVOE = 1 **Note:** Only accessible in host mode.
4	AVOE	Override enable of A-peripheral session valid 0: Override is disable. Internally A-peripheral session valid received from PHY is selected 1: Override is enable. Internally A-peripheral session valid received from PHY is overridden with AVOV **Note:** Only accessible in host mode.
3	VOV	Override value of VBUS valid 0: VBUS valid value is 0 when VOE = 1 1: VBUS valid value is 1 when VOE = 1 **Note:** Only accessible in host mode.
2	VOE	Override enable of VBUS valid 0: Override is disable. Internally VBUS valid received from PHY is selected 1: Override is enable. Internally VBUS valid received from PHY is overridden with VOV **Note:** Only accessible in host mode.
1	SRPREQ	SRP request This bit is set by software to start a SRP on the USB. Software can clear this bit when SRPEND bit in USBHS_GOTGINTF register is set, by writing zero to it, or clearing the SRPEND bit in USBHS_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 successes and cleared when SRPREQ bit is set. 0: SRP fails

1: SRP successes

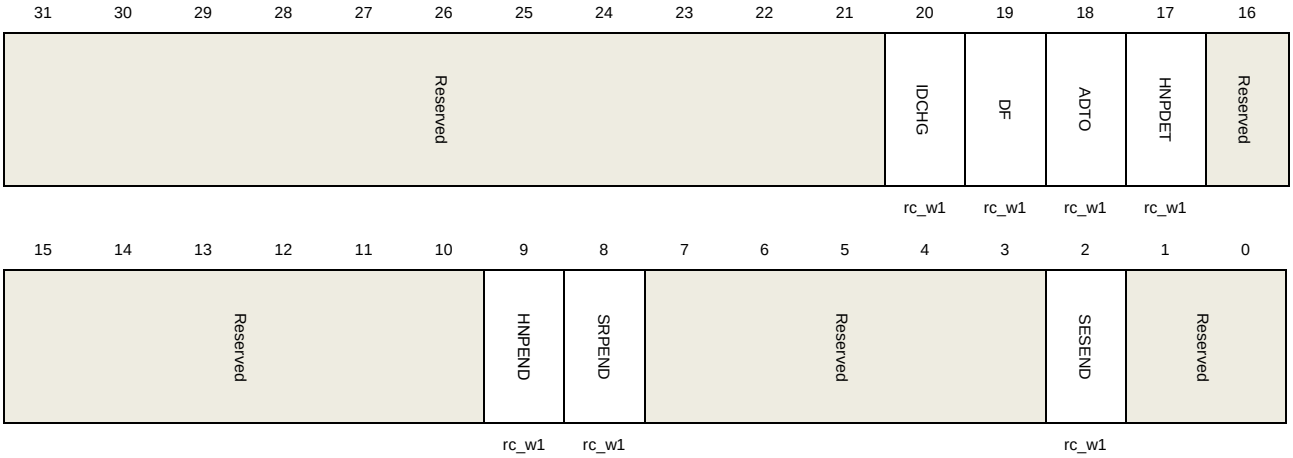
Note: Only accessible in device mode.

Global OTG interrupt flag register (USBHS_GOTGINTF)

Address offset: 0x0004

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20	IDCHG	There is a change in the value of ID input
19	DF	Debounce finish Set by USBHS when the debounce during device connection is done. **Note:** Only accessible in host mode.
18	ADTO	A-device timeout Set by USBHS 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 USBHS 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. Software should read the HNPS in USBHS_GOTGCS register to get the result of HNP. **Note:** Accessible in both device and host modes.
8	SRPEND	SRPEND

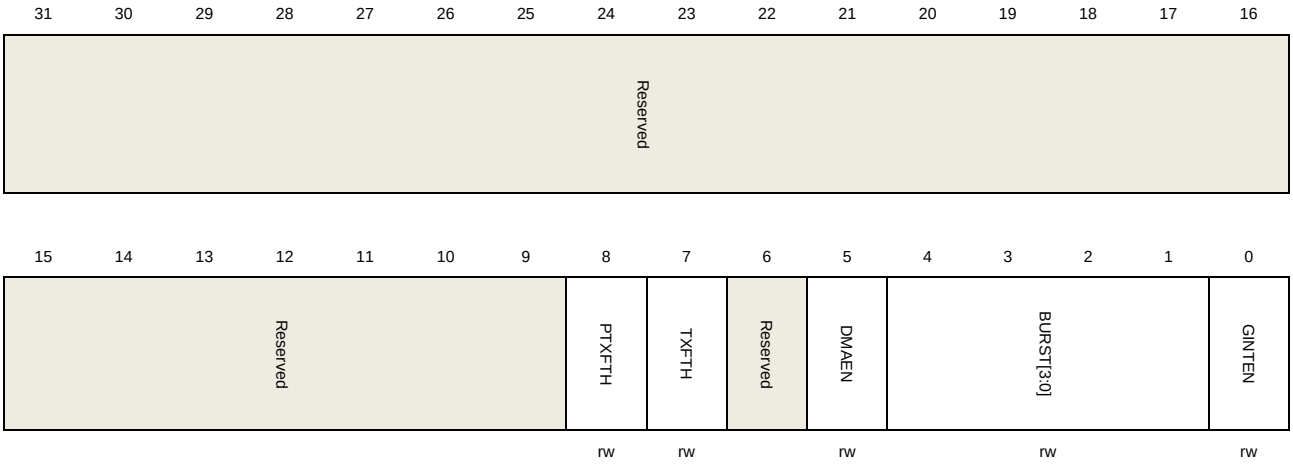
		Set by the core when a SRP ends. Software should read the SRPS in USBHS_GOTGCS register to get the result of SRP. **Note:** Accessible in both device and host modes.
7:3	Reserved	Must be kept at reset value.
2	SESEND	Session end Set by the core when VBUS voltage is below Vb_ses_vld.
1:0	Reserved	Must be kept at reset value.

Global AHB control and status register (USBHS_GAHBCS)

Address offset: 0x0008

Reset value: 0x0000 0000

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



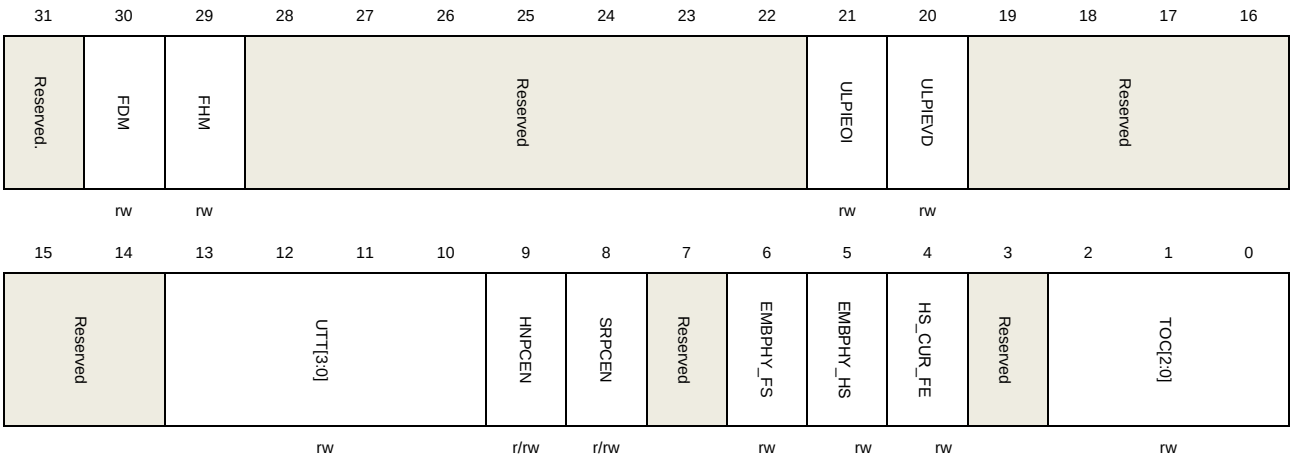
Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	PTXFTH	Periodic Tx FIFO threshold 0: PTXFEIF will be triggered when the periodic transmit FIFO is half empty 1: PTXFEIF will be triggered when the periodic transmit FIFO is completely empty **Note:** Only accessible in host mode.
7	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	Reserved	Must be kept at reset value.
5	DMAEN	DMA function Enable 0: DMA function is disabled 1: DMA function is enabled
4:1	BURST[3:0]	The AHB burst type used by DMA 0000: Single 0001: INCR 0011: INCR4 0101: INCR8 0111: INCR16
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 (USBHS_GUSBCS)

Address offset: 0x000C
Reset value: 0x0000 0A00

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



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 USBHS 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 USBHS 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.
28:22	Reserved	Must be kept at reset value.
21	ULPIEOI	ULPI external over-current indicator ULPI PHY uses this bit to decide whether to use internal or external over-current indicator. This bit only takes effect when external ULPI PHY is used (EMBPHY_FS and EMBPHY_HS bits in this register are both 0). 0: ULPI PHY uses internal over-current indicator 1: ULPI PHY uses external over-current indicator
20	ULPIEVD	ULPI external VBUS driver ULPI PHY uses this bit to decide whether VBUS is driven by ULPI PHY or by external power supply. This bit only takes effect when external ULPI PHY is used (EMBPHY_FS and EMBPHY_HS bits in this register are both 0). 0: VBUS is driven by ULPI PHY 1: VBUS is driven by external power supply
19:14	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	Reserved	Must be kept at reset value.
6	EMBPHY_FS	Embedded FS PHY selected

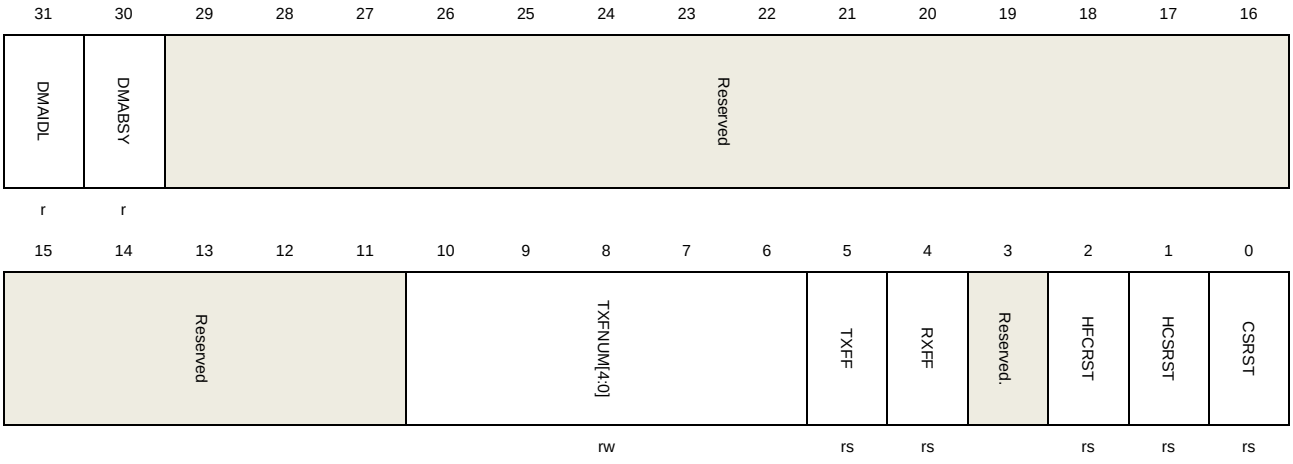
		0: Embedded FS PHY is disabled 1: Embedded FS PHY is enabled **Note:** This bit can only be set when EMBPHY_HS is set to 0. Accessible in both device and host modes.
5	EMBPHY_HS	Embedded HS PHY selected 0: Embedded HS PHY is disabled 1: Embedded HS PHY is enabled **Note:** This bit can only be set when EMBPHY_FS is set to 0. Accessible in both device and host modes.
4	HS_CUR_FE	HS current software enable 0: Release the HS mode TX current enable 1: Force HS mode TX current enable
3	Reserved	Must be kept at reset value.
2:0	TOC[2:0]	Timeout calibration USBHS 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 decided by which PHY is used: 48MHZ with internal embedded PHY and 60MHz with external ULPI PHY.)

Global reset control register (USBHS_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)



Bits	Fields	Descriptions
31	DMAIDL	DMA Idle state This bit reports that whether DMA is in IDLE state or not.

		0: DMA is not IDLE 1: DMA is IDLE **Note:** Accessible in both device and host modes.
30	DMABSY	DMA Busy This bit reports that whether DMA is busy. 0: DMA is not busy 1: DMA is busy **Note:** Accessible in both device and host modes.
29:11	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 … 00101: Only Tx FIFO5 is flushed 1XXXX: All Tx FIFOs are flushed Other: Non data FIFO is flushed
5	TXFF	Tx FIFO flush Application sets 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 USBHS. **Note:** Accessible in both device and host modes.
4	RXFF	Rx FIFO flush Application sets 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 USBHS. **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 USBHS. 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 USBHS.

		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 USBHS. **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 (USBHS_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	LPMIF	PTXFEIF	HCIF	HPIF	Reserved		PXNCIF/ ISOONCIF	ISOINCIF	OEPIF	IEPIF	Reserved	
rc_w1	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	LPMIF	LPM interrupt flag In host mode, when device responds to LPM transaction with ACK, NYET or STALL, or when host has sent RECNT (USBHS_LPMCFG register) times LPM transaction, the interrupt is triggered. In device mode, when device has received LPM transaction and responds with ACK, NYET or STALL, the interrupt is triggered.
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 USBHS_GAHBCS register. **Note:** Only accessible in host mode.
25	HCIF	Host channels interrupt flag Set by USBHS when one of the channels in host mode has raised an interrupt. Software should first read USBHS_HACHINT register to get the channel number, and then read the corresponding USBHS_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 USBHS detects that port status changes in host mode. Software should read USBHS_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 USBHS 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 USBHS_DCFG), USBHS will set this bit if there are still isochronous OUT endpoints 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 USBHS_DCFG), USBHS will set this bit if there are still isochronous IN endpoints that not completed transactions. (Device Mode) **Note:** Only accessible in device mode.

19	OEPIF	OUT endpoint interrupt flag Set by USBHS when one of the OUT endpoints in device mode has raised an interrupt. Software should first read USBHS_DAEPINT register to get the device number, and then read the corresponding USBHS_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 USBHS when one of the IN endpoints in device mode has raised an interrupt. Software should first read USBHS_DAEPINT register to get the device number, and then read the corresponding USBHS_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 has reaches the value defined by EOPFT[1:0] bits in USBHS_DCFG register, USBHS sets this flag. **Note:** Only accessible in device mode.
14	ISOOPDIF	Isochronous OUT packet dropped interrupt flag USBHS 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 USBHS sets this bit after the speed enumeration finishes. Software is able to read USBHS_DSTAT register to get the current device speed. **Note:** Only accessible in device mode.
12	RST	USB reset USBHS sets this bit when it detects a USB reset signal on bus. **Note:** Only accessible in device mode.
11	SP	USB suspend USBHS 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 USBHS 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

		Software is able to write 1 to SGONAK bit in the USBHS_DCTL register and USBHS will set GONAK flag after writing to SGONAK takes effect. And this bit can be cleared by writting 1 to CGONAK bit in the USBHS_DCTL register. **Note:** Only accessible in device mode.
6	GNPINAK	Global IN Non-Periodic NAK effective Software is able to write 1 to SGINAK bit in the USBHS_DCTL register and USBHS will set GNPINAK flag after writing to SGINAK takes effect. And this bit can be cleared by writting 1 to CGINAK bit in the USBHS_DCTL register. **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 USBHS_GAHBCS register. **Note:** Only accessible in host mode.
4	RXFNEIF	Rx FIFO non-empty interrupt flag USBHS 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: USBHS 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: USBHS sets this bit to 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 USBHS sets this bit when the flags in USBHS_GOTGINTF register generate a interrupt. Software should read USBHS_GOTGINTF register to get the source of this interrupt. This bit is cleared after the flags in USBHS_GOTGINTF causing this interrupt are cleared. **Note:** Accessible in both host and device modes.
1	MFIF	Mode fault interrupt flag USBHS sets this bit if 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 (USBHS_GINTEN)

Address offset: 0x0018

Reset value: 0x0000 0000

This register works with the global interrupt flag register (USBHS_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	LPMIE	PTXFEIE	HCIE	HPIE	Reserved		PXNCIE/ ISOONCIE	ISOINCIE	OEPIE	IEPIE	Reserved	
rw	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	LPMIE	LPM interrupt enable 0: disable LPM interrupt

		1: enable LPM interrupt **Note:** Accessible in both host and device modes.
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 (USBHS_GRSTATR/USBHS_GRSTATP)

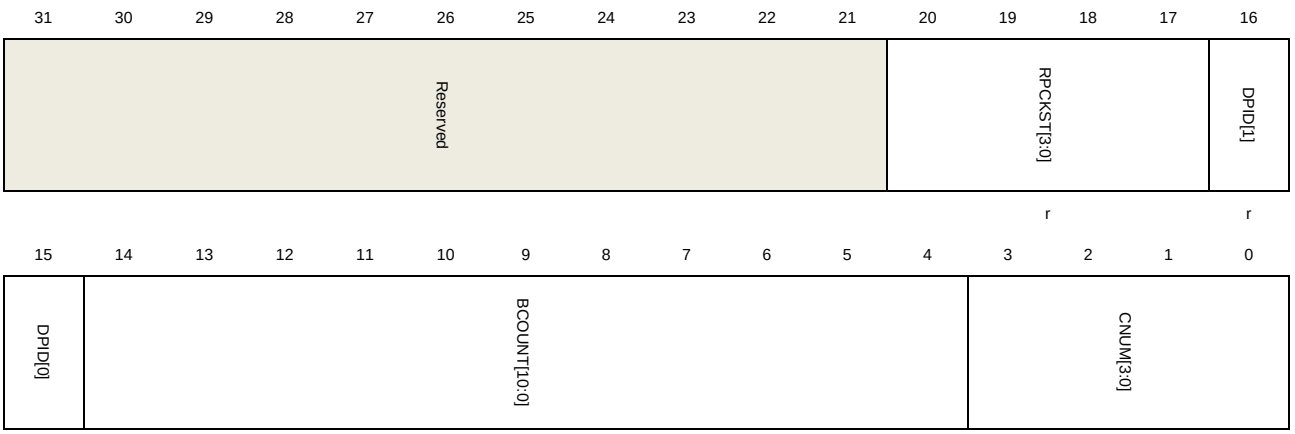
Address offset for Read: 0x001C
Address offset for Pop: 0x0020
Reset value: 0x0000 0000

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 USBHS_GINTF) is triggered.

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

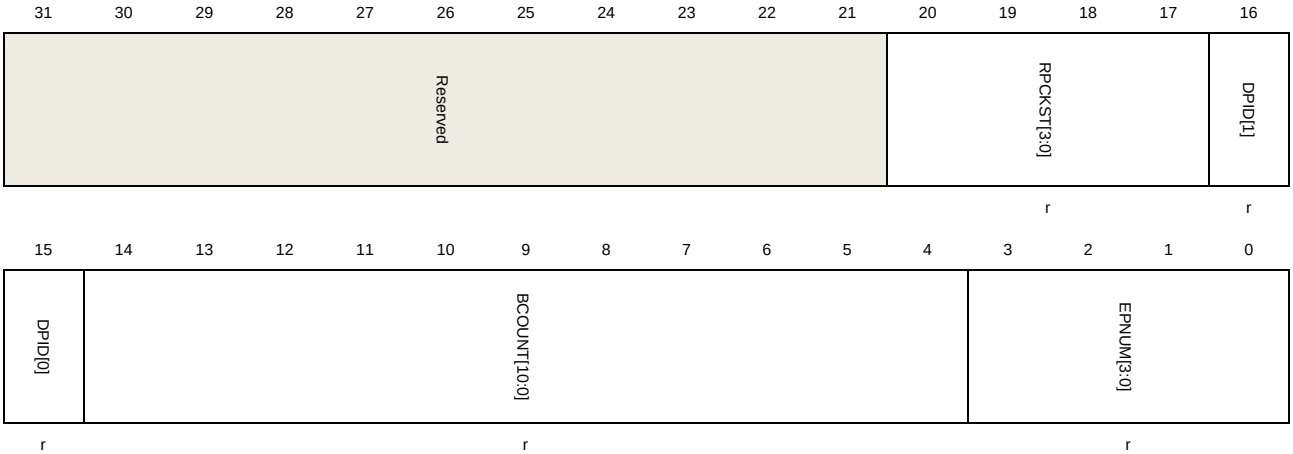
Host mode:



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:



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)

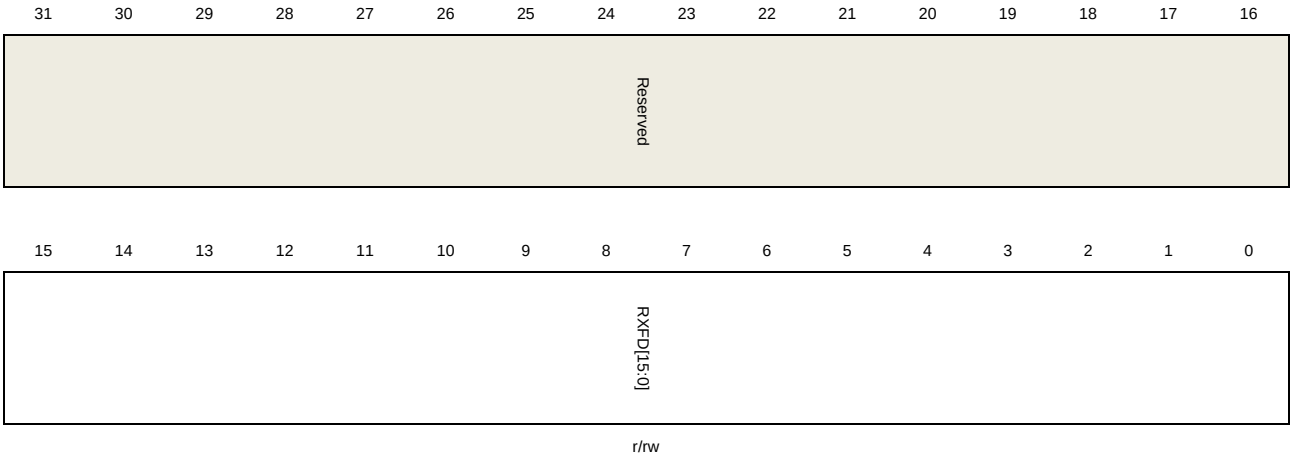
		0100: SETUP transaction completed (generates an interrupt) 0110: SETUP data packet received Others: Reserved
16:15	DPID[1:0]	Data PID The Data PID of the received OUT data packet 00: DATA0 10: DATA1 01: DATA2 11: MDATA
14:4	BCOUNT[10:0]	Byte count The byte count of the received data packet.
3:0	EPNUM[3:0]	Endpoint number The endpoint number to which the current received packet belongs.

Global receive FIFO length register (USBHS_GRFLEN)

Address offset: 0x024

Reset value: 0x0000 0200

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

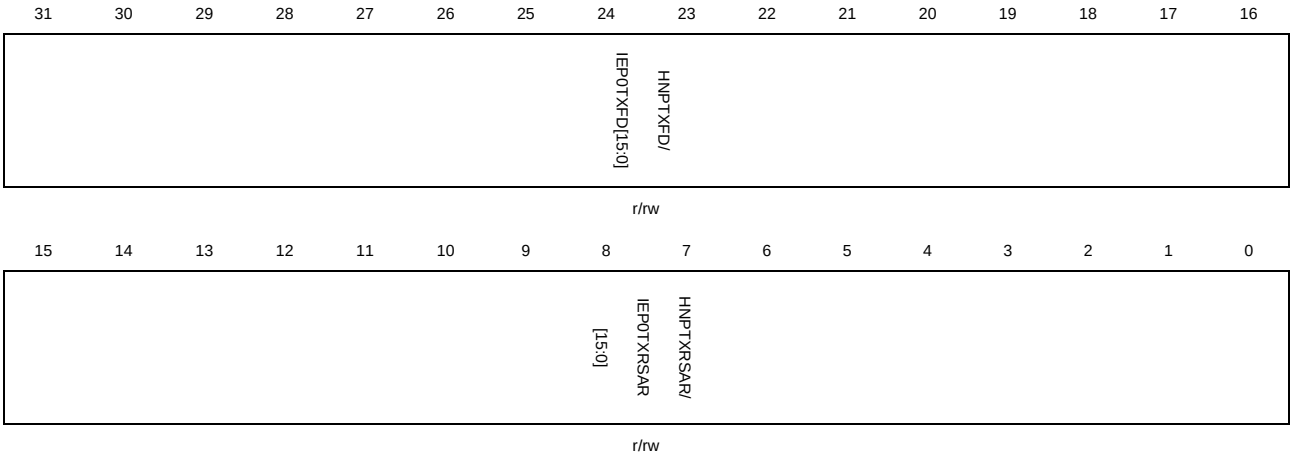


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	RXFD[15:0]	Rx FIFO depth In terms of 32-bit word. 1≤RXFD≤1024

Host non-periodic transmit FIFO length register /Device IN endpoint 0 transmit FIFO length (USBHS_HNPTFLEN _DIEP0TFLEN)

Address offset: 0x028
Reset value: 0x0200 0200

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



Host Mode:

Bits	Fields	Descriptions
31:16	HNPTXFD[15:0]	Non-periodic Tx FIFO depth In terms of 32-bit word. 1≤HNPTXFD≤1024
15:0	HNPTXRSAR[15:0]	Non-periodic Tx RAM start address The start address for non-periodic transmit FIFO RAM in terms of 32-bit word.

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 in terms of 32-bit word.

Host non-periodic transmit FIFO/queue status register (USBHS_HNPTFQSTAT)

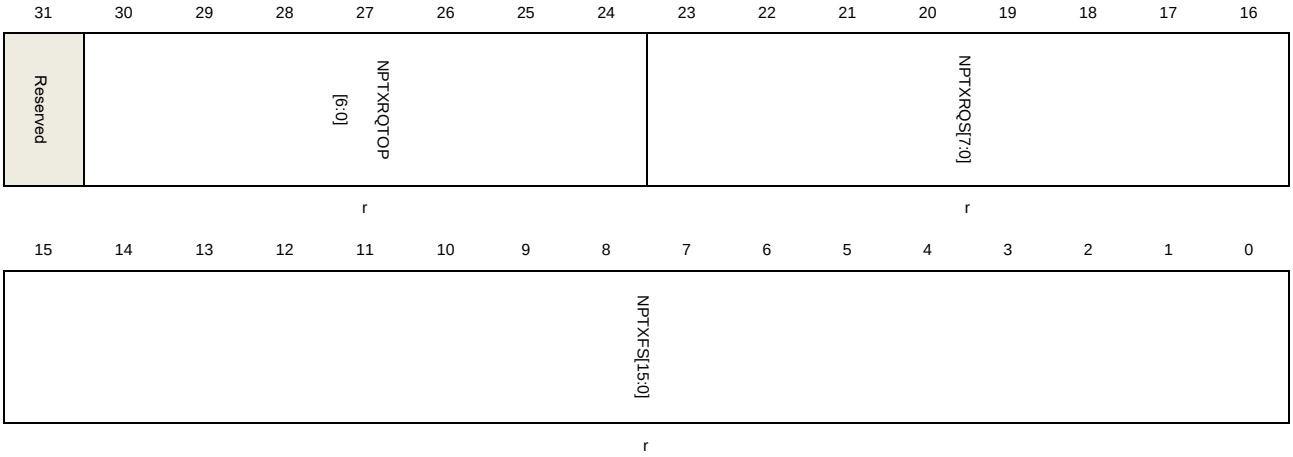
Address offset: 0x002C
Reset value: 0x0008 0200

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

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.



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 words 2: 2 words … n: n words (0≤n≤NPTXFD)

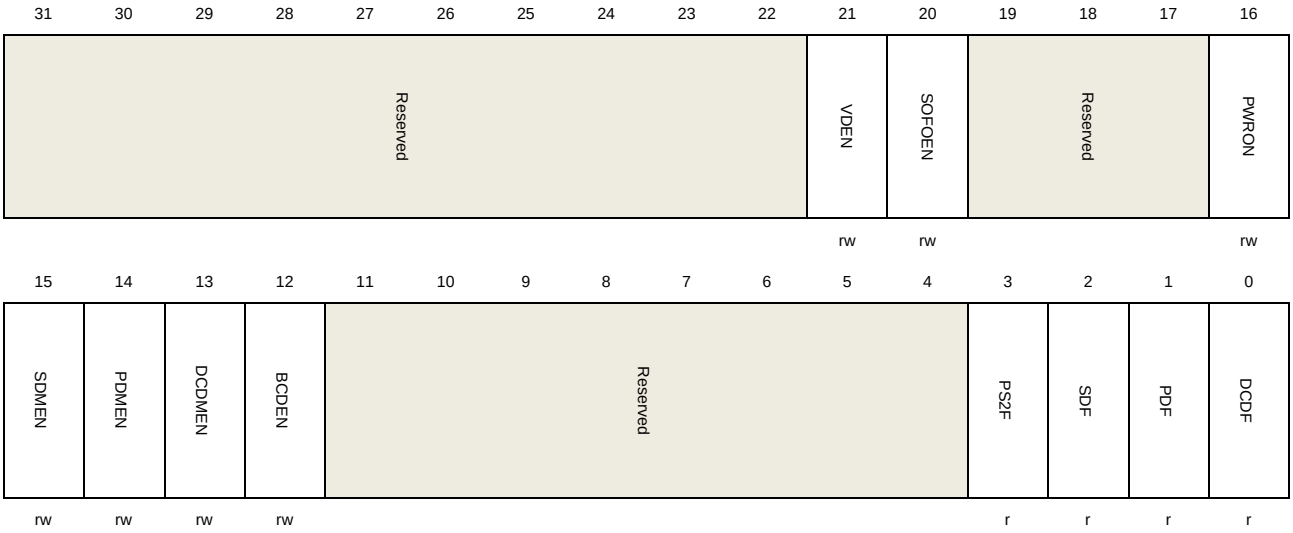
Others: Reserved

Global core configuration register (USBHS_GCCFG)

Address offset: 0x0038

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value.
21	VDEN	Enable of VBUS sensing comparator to detect VBUS valid. VBUS comparator is enabled automatically if HNP or SRP is supported. 0: VBUS detection is disabled 1: VBUS detection is enabled
20	SOFOEN	SOF output enable 0: SOF pulse output disabled. 1: SOF pulse output enabled.
19:17	Reserved	Must be kept at reset value.
16	PWRON	Power on This bit is the power switch for the internal embedded PHY. 0: Embedded PHY power off. 1: Embedded PHY power on.
15	SDMEN	Secondary detection mode enable 0: Secondary detection disable 1: Secondary detection enable
14	PDMEN	Primary detection mode enable 0: Primary detection disable

		1: Primary detection enable
13	DCDMEN	Data connect detection mode enable 0: Data connect detection disable 1: Data connect detection enable
12	BCDEN	Battery charging detection enable 0: Battery charging detection disable 1: Battery charging detection enable
11:4	Reserved	Must be kept at reset value.
3	PS2F	PS2 detection status, it is active only in Primary detection mode 0: Normal port is detected 1: PS2 port is detected
2	SDF	Secondary detection status 0: CDP is detected 1: DCP is detected
1	PDF	Primary detection status 0: no BCD supported is detected 1: BCD supported is detected
0	DCDF	Data connect detection status 0: Data line connect is not detected 1: Data line connect is detected

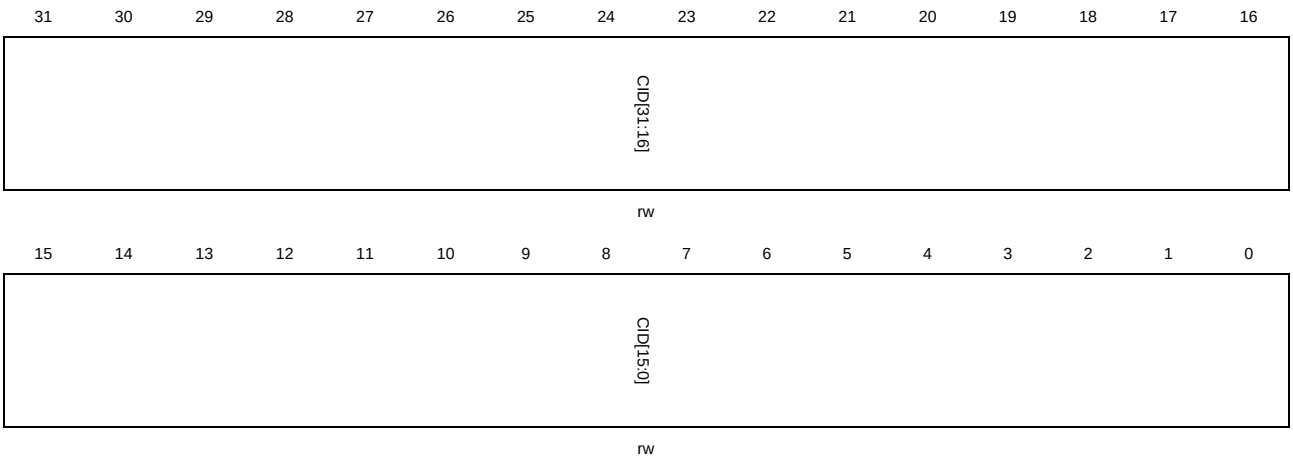
Core ID register (USBHS_CID)

Address offset: 0x003C

Reset value: 0x0000 1000

This register contains the Product ID.

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



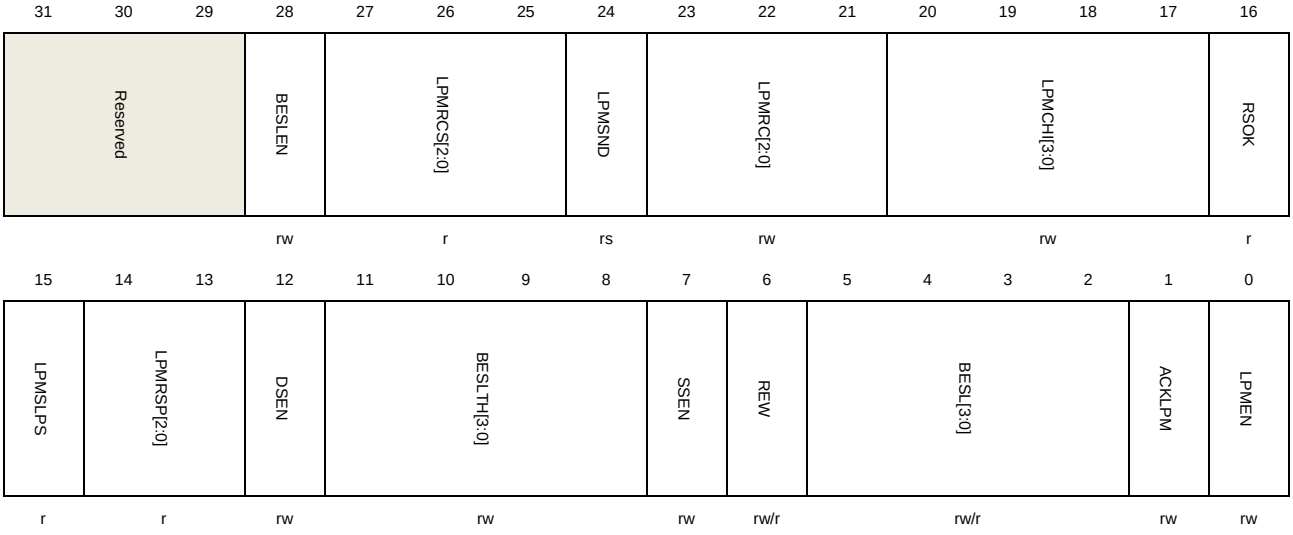
Bits	Fields	Descriptions
31:0	CID[31:0]	Core ID Software can write or read this field and uses this field as a unique ID for its application.

Global core LPM configuration register (USBHS_GLPMCFG)

Address offset: 0x0054

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value.
28	BESLEN	LPM Errata selection enable 0: USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification is selected 1: Errata for USB 2.0 ECN: Link Power Management (LPM) is selected
27:25	LPMRCS[2:0]	LPM retry count status **Note:** Only accessible in host mode
24	LPMSND	Send LPM transaction When ACK, STALL or NYET response is received, or all the LPM of retry count have been sent, the hardware clears this bit. **Note:** Only accessible in host mode
23:21	LPMRC[2:0]	LPM retry count It is the number of retry count when an ERROR response is received, until ACK, STALL or NYET response is received **Note:** Only accessible in host mode

20:17	LPMCHI[3:0]	Channel number index when send LPM transaction **Note:** Only accessible in host mode
16	RSOK	Resume can be sent after sleep state Host or device can send resume from sleep state after 50us (TL1Residency). When LPMSLPS is 0, it is reset. 1: Resume can be started from sleep state 0: Resume cannot be started from sleep state
15	LPMSLPS	Sleep status **Host mode:** The host transitions to sleep status after receiving ACK response. **Device mode:** The device enters into sleep status after sending ACK response and the TL1TokenRetry timer has been expired. 1: Core is in sleep status 0: Core is not in sleep status
14:13	LPMRSP[1:0]	Response of LPM 11: ACK 10: NYET 01: STALL 00: ERROR (no response)
12	DSEN	Deep sleep enable Enable suspending the PHY in deep sleep mode
11:8	BESLTH[3:0]	BESL threshold **Device mode:** When BESL is greater than or equal to the BESLTH value, device enters into deep low power mode. **Host mode:** BESLTH indicates the duration of resume signal (TL1HubDrvResume2) when it detects device initialed resume. 0000: 75us 0001: 100us 0010: 150us 0011: 250us 0101: 450us 0110: 950us
7	SSEN	Shallow sleep enable Enable suspending the PHY in shallow sleep mode
6	REW	bRemoteWake value **Host mode:** The remote wake up value to be sent in LPM transaction

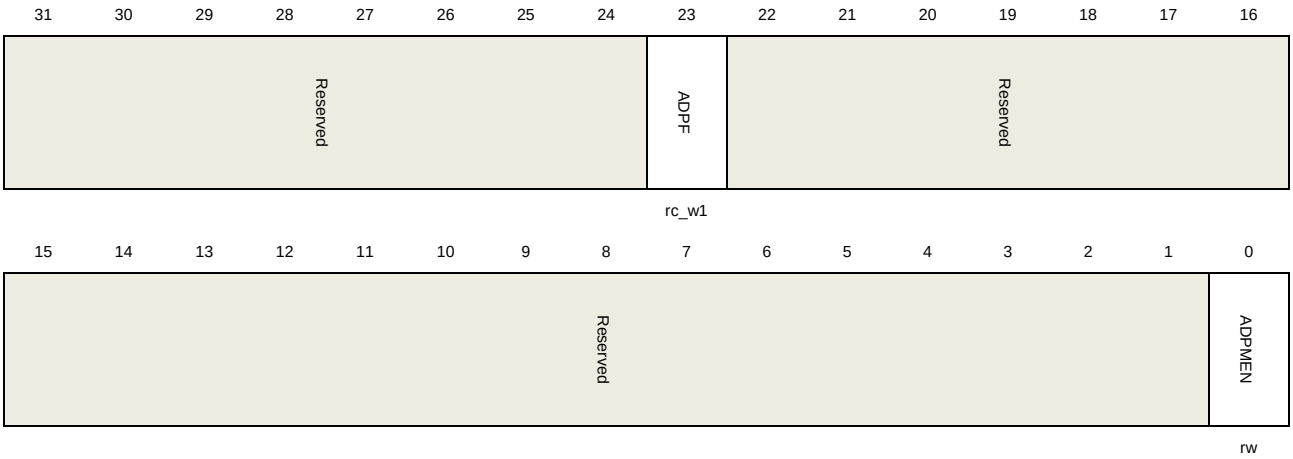
		Device mode (read-only): When ACK, STALL or NYET is sent, it is updated with bRemoteWake value in received LPM transaction
5:2	BESL[3:0]	Best effort service latency **Host mode:** BESL value to be sent in LPM transaction. It is also the duration of resume (TL1HubDrvResume1) when host initialed resume. **Device mode:** When ACK, STALL or NYET is sent, it is updated with BESL value in received LPM transaction 0000: 125us 0001: 150us 0010: 200us 0011: 300us 0100: 400us 0101: 500us 0110: 1000us 0111: 2000us 1000: 3000us 1001: 4000us 1010: 5000us 1011: 6000us 1100: 7000us 1101: 8000us 1110: 9000us 1111: 10000us
1	ACKLPM	ACK in LPM transaction enable 1: ACK The device response with ACK only on successful LPM transaction. - No ERROR in LPM transaction - No data pending error - bLinkState = 0001 in received LPM transaction 0: NYET The device response with NYET - The received bLinkState value is not 0001 - There is an error in received LPM transaction **Note:** Only accessible in device mode
0	LPMEN	LPM enable 1: Enable LPM 0: Disable LPM

Power down register (USBHS_PWRD)

Address offset: 0x0058

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23	ADPF	ADP event interrupt flag
22:1	Reserved	Must be kept at reset value.
0	ADPMEN	ADP module enable 1: ADP module is enable 0: ADP module is disable

ADP control and status register (USBHS_ADPCTL)

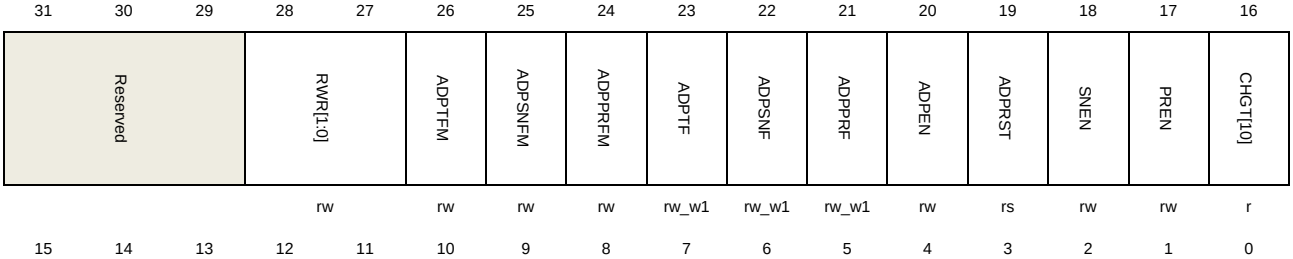
Address offset: 0x0060

Reset value: 0x0000 0000

In order to write in the register, program RWR with 10 and keep polling until RWR = 00.

In order to read from the register, wait any ADP flag is set or write RWR with 01 and keep polling until RWR = 00.

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



CHGT[9:0]	PERPR[1:0]	RESOPR[1:0]	DSCHGPR[1:0]
r	rw	rw	rw

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value.
28:27	RWR[1:0]	Read and write request 00: Read or write valid (updated by core) 01: Read request 10: Write request
26	ADPTFM	The mask of ADP timeout interrupt flag
25	ADPSNFM	The mask of ADP sense interrupt flag
24	ADPPRFM	The mask of ADP probe interrupt flag
23	ADPTF	ADP timeout interrupt flag
22	ADPSNF	ADP sense interrupt flag
21	ADPPRF	ADP probe interrupt flag
20	ADPEN	ADP enable 1: ADP is enable 0: ADP is disable
19	ADPRST	ADP reset This is cleared automatically by core after reset procedure
18	SNEN	ADP sense enable 1: Sense is enable 0: Sense is disable
17	PREN	ADP probe enable 1: Probe is enable 0: Probe is disable
16:6	CHGT[10:0]	The latest time that VBUS ramps from VADPSINK to VADPPRB. These bits are defined in units of 32 kHz clock cycle. 000: 1 cycle 001: 2 cycles 002: 3 cycles 003: 4 cycles … 7ff: 2048 cycles

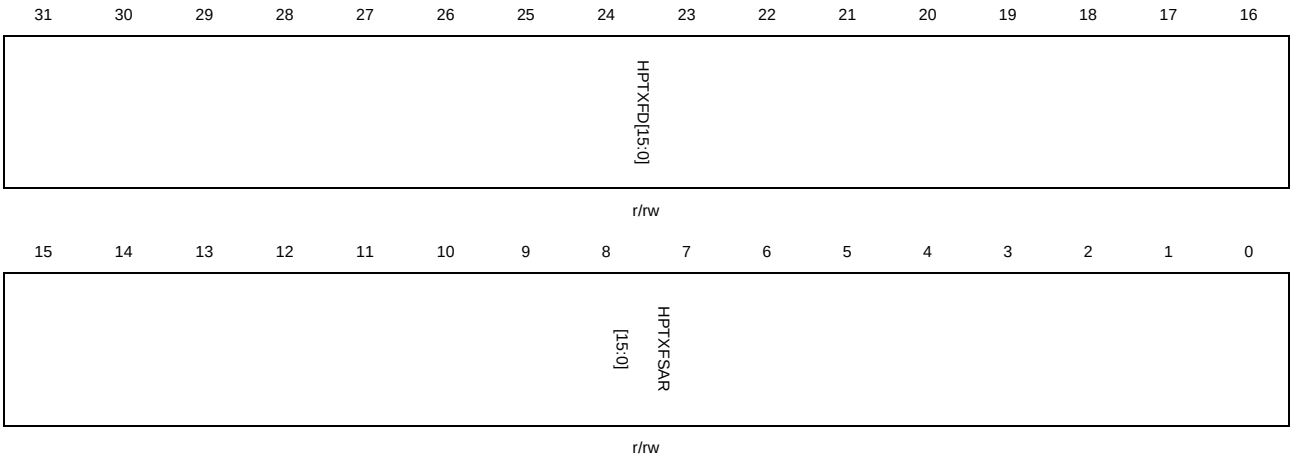
5:4	PERPR[1:0]	Period of probe 00: 0.625 to 0.925 second 01: 1.25 to 1.85 second 10: 1.9 to 2.6 second
3:2	RESOPR[1:0]	The resolution of CHGT value. These bits are defined in units of 32 kHz clock cycle. If 10 is chosen, the CHGT increments for every three 32 kHz clock cycle. 00: 1 cycle 01: 2 cycles 10: 3 cycles 11: 4 cycles
1:0	DSCHGPR[1:0]	Time of probe discharge 00: 4 ms 01: 8 ms 10: 16 ms 11: 32 ms

Host periodic transmit FIFO length register (USBHS_HPTFLEN)

Address offset: 0x0100

Reset value: 0x0200 0600

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

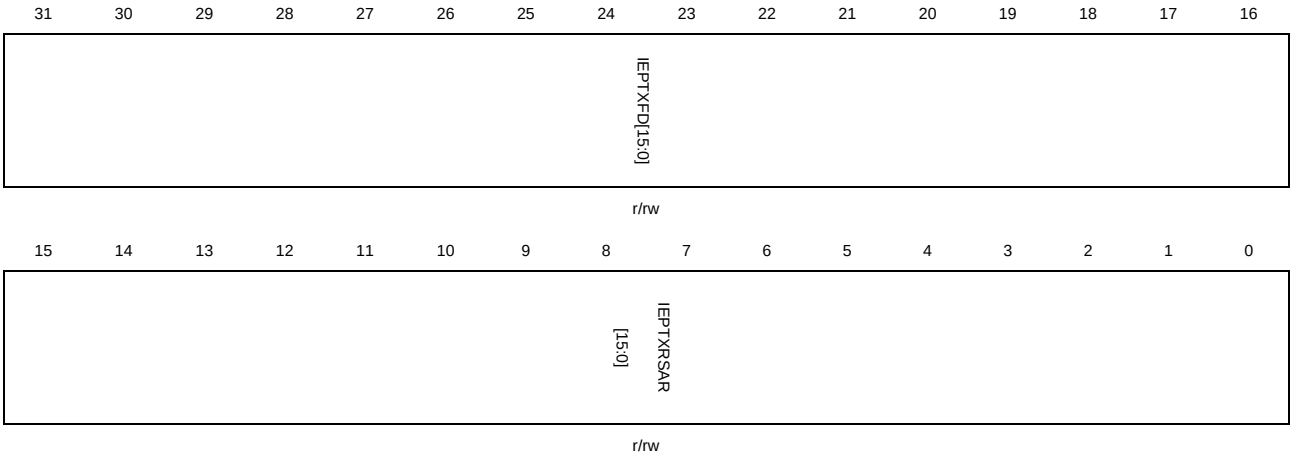


Bits	Fields	Descriptions
31:16	HPTXFD[15:0]	Host Periodic Tx FIFO depth In terms of 32-bit word. 1≤HPTXFD≤1024
15:0	HPTXFSAR[15:0]	Host periodic Tx RAM start address The start address for host periodic transmit FIFO RAM in terms of 32-bit word.

Device IN endpoint transmit FIFO length register (USBHS_DIEPxTFLEN) (x = 1..5, 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)



Bits	Fields	Descriptions
31:16	IEPTXFD[15:0]	IN endpoint Tx FIFO x depth In terms of 32-bit word. 1≤IEPTXFD≤1024
15:0	IEPTXRSAR[15:0]	IN endpoint FIFOx Tx x RAM start address The start address for IN endpoint transmit FIFO x in terms of 32-bit word.

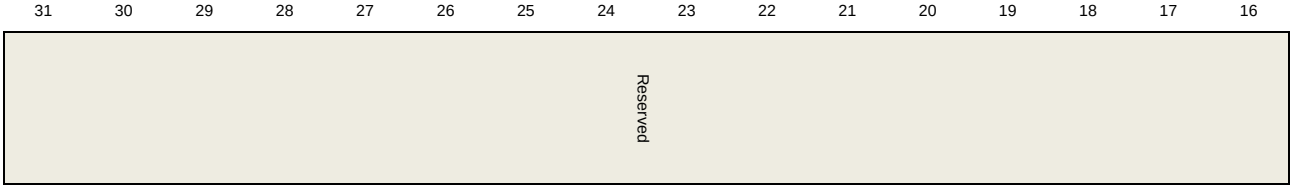
29.7.2. Host control and status registers

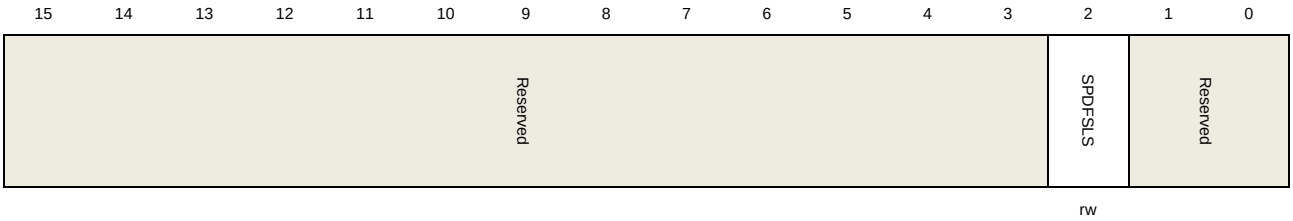
Host control register (USBHS_HCTL)

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)





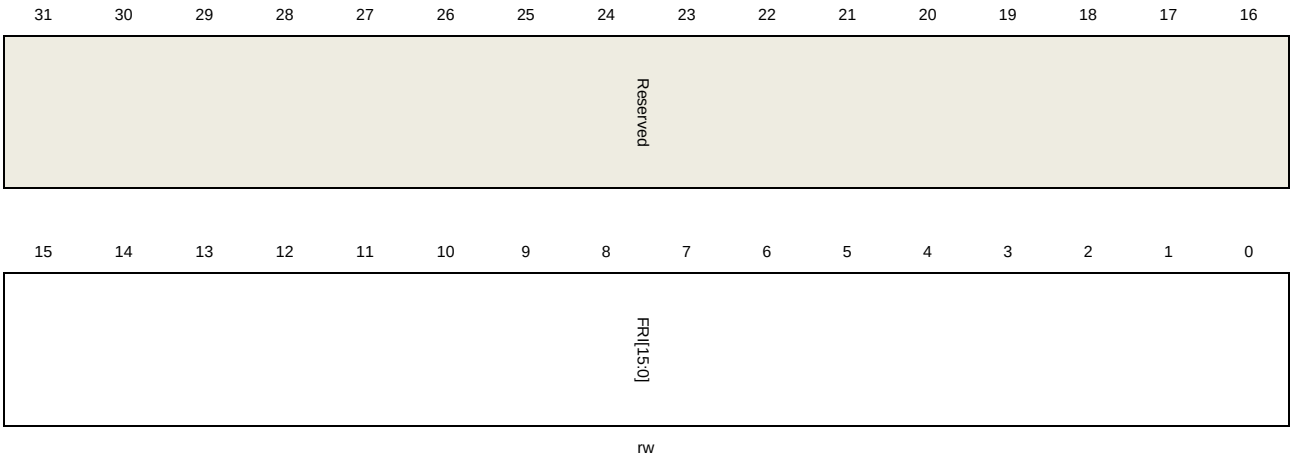
Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2	SPDFSLS	Speed limited to FS and LS Software may use this bit to limit USBHS's enumeration speed to FS/LS and make USBHS not perform High-Speed enumeration during reset. 0: Speed not limited. 1: Speed limited in FS/LS only.
1:0	Reserved	Must be kept at reset value.

Host frame interval register (USBHS_HFT)

Address offset: 0x0404
Reset value: 0x0000 BB80

This register sets the frame interval for the current enumerating speed when USBHS controller is enumerating.

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



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 clock. Port is enabled after a port reset operation, USBHS uses 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:

Internal Embedded PHY:

High-Speed: 60MHz

Full-Speed: 48MHz

Low-Speed: 6MHz

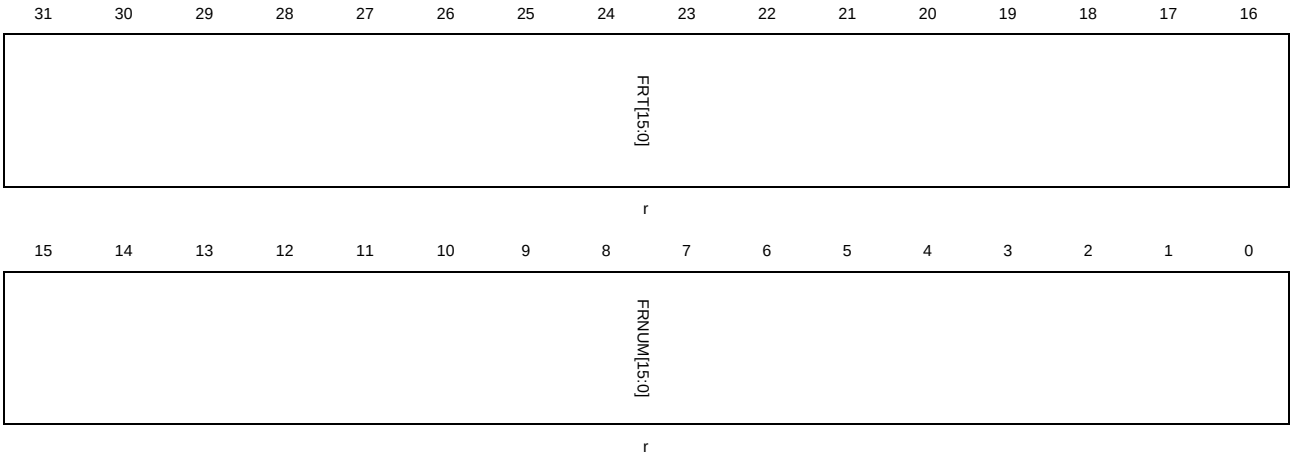
External ULPI PHY: 60MHz

Host frame information remaining register (USBHS_HFINFR)

Address offset: 0x408

Reset value: 0xBB80 0000

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



Bits	Fields	Descriptions
31:16	FRT[15:0]	Frame remaining time This field reports the remaining time of current frame in terms of PHY clock.
15:0	FRNUM[15:0]	Frame number This field reports the frame number of current frame and returns to 0 after it reaches 0x3FF.

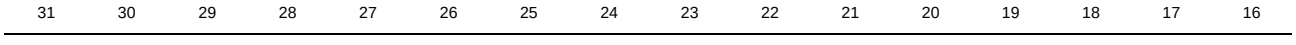
Host periodic transmit FIFO/queue status register (USBHS_HPTFQSTAT)

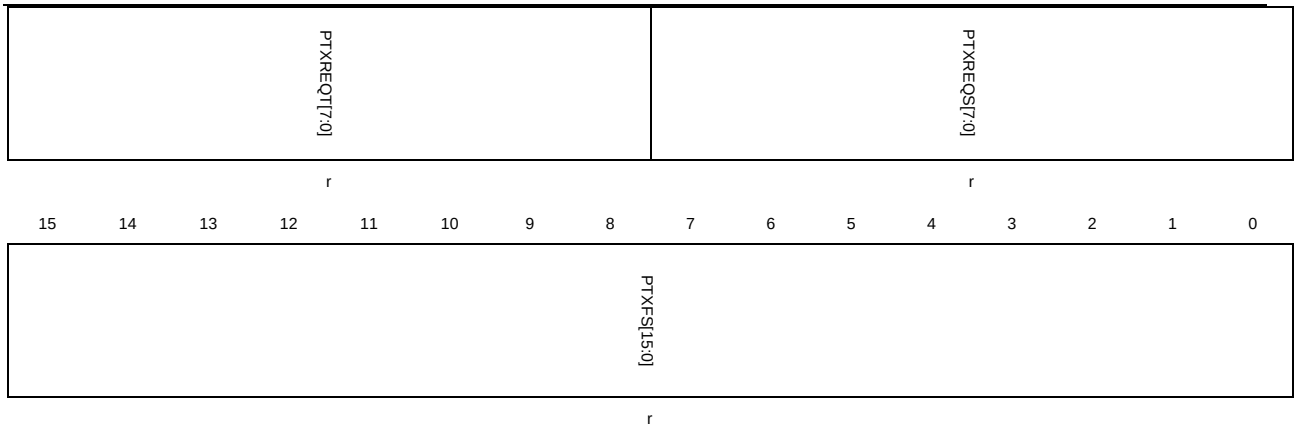
Address offset: 0x0410

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)





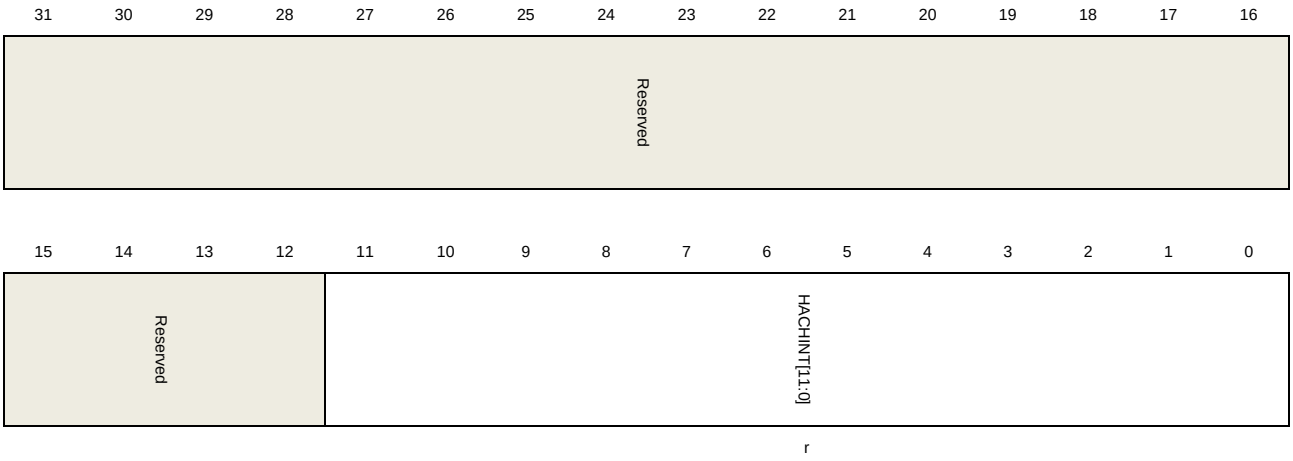
Bits	Fields	Descriptions
31:24	PTXREQT[7:0]	Top entry of the periodic Tx request queue Entry in the periodic transmit request queue. Bit 31: Odd/Even frame – 0: send in even frame – 1: send in odd frame Bits 30:27: Channel Number Bits 26:25: – 00: IN/OUT token – 01: Zero-length OUT packet – 11: Channel 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 word. 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 (USBHS_HACHINT)

Address offset: 0x0414
Reset value: 0x0000 0000

When a channel interrupt is triggered, USBHS sets corresponding bit in this register and software should read this register to know which channel is asserting interrupt.

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



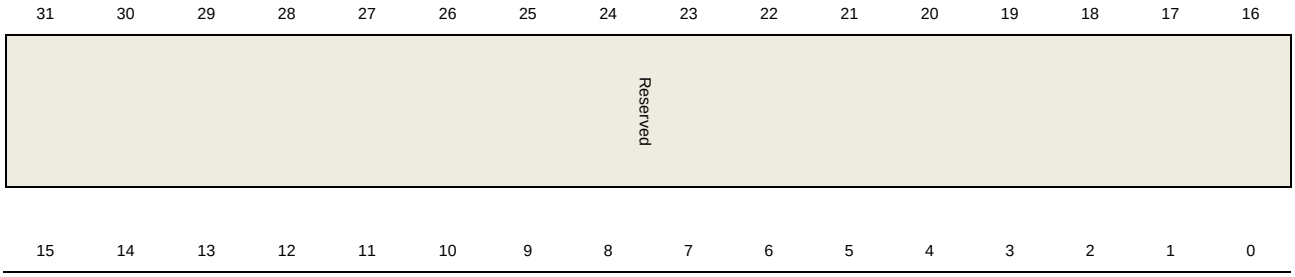
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	HACHINT[11:0]	Host all channel interrupts Each bit represents a channel: Bit 0 for channel 0, bit 11 for channel 11.

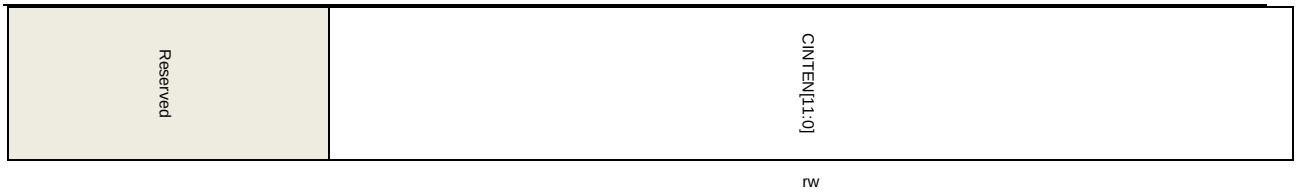
Host all channels interrupt enable register (USBHS_HACHINTEN)

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 USBHS_GINTF register.

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





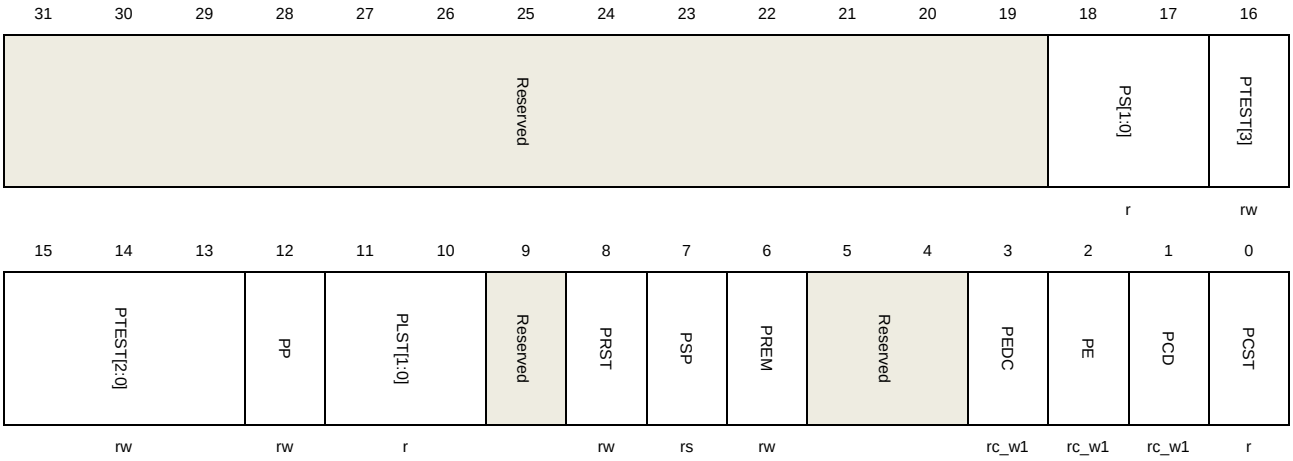
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	CINTEN	Channel interrupt enable 0: Disable channel-n interrupt 1: Enable channel-n interrupt Each bit represents a channel: Bit 0 for channel 0, bit 11 for channel 11.

Host port control and status register (USBHS_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 USBHS_GINTF register will be triggered if one of these flags in this register is set by USBHS: PRST, PEDC and PCD.

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



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. 00: High-Speed 01: Full-Speed 10: Low-Speed Others: Reserved

16:13	PTEST[3:0]	Port Test control The software writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is sended on the port. When test mode is used, The HS_CUR_FE bit in USBHS_GUSBCS register should also be set. 0000: Test mode disabled 0001: Test_J mode 0010: Test_K mode 0011: Test_SE0_NAK mode 0100: Test_Packet mode 0101: Test_Force_Enable Others: Reserved
12	PP	Port power This bit should be set before a port is used. Because USBHS 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 disconnection 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 When the application sets PREM in sleep status, the core continues to drive the resume signal until the timer specified with BESLTH has expired. When the core detects a USB remote wakeup, it starts driving the resume signal and clears it automatically at the end of resume.
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 USBHS after a USB reset signal finishes and cannot be set by software. This bit is cleared by the following events: – A disconnection condition – Software clears this bit 0: Port disabled 1: Port enabled
1	PCD	Port connect detected Set by USBHS 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 (USBHS_HCHxCTL) (x = 0..11, 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]							MPC [1:0]		EPTYPE[1:0]		LSD	Reserved
rs	rs	rw	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 USBHS. 0: Channel disabled 1: Channel enabled Software should follow 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	MPC[1:0]	Multiple Packet Count For periodic transfers, this field indicates to the number of transactions that must be issued per micro-frame. For nonperiodic transfers, it defines how many packets the DMA should fetch or write for this channel before the internal DMA engine changes arbitration. 00: Reserved 01: 1 transaction to be issued per micro-frame 10: 2 transactions to be issued per micro-frame 11: 3 transactions to be issued per micro-frame
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.

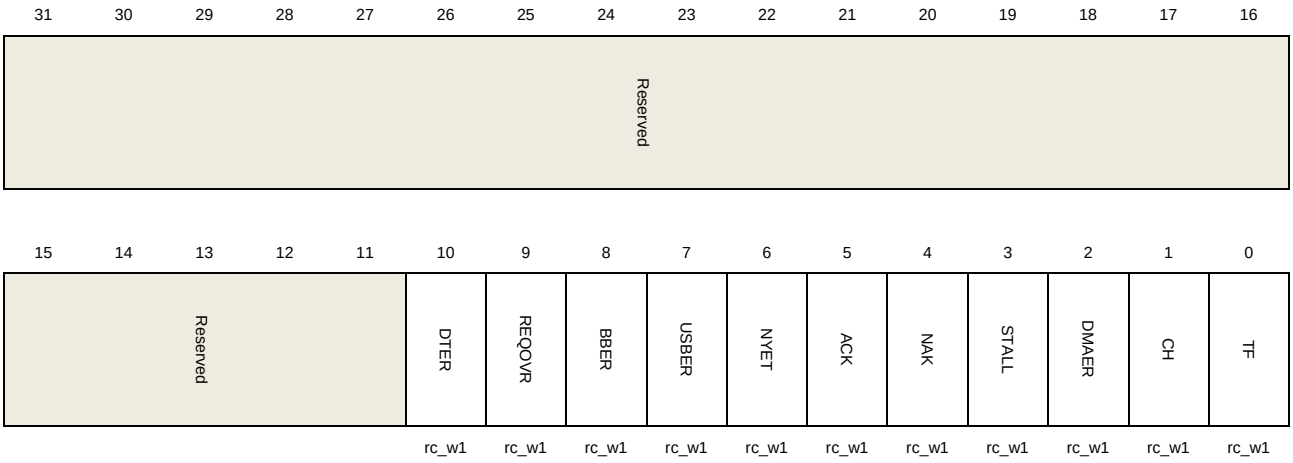
16	Reserved	Must be kept at reset value.
15	EPDIR	Endpoint direction The transfer direction of the endpoint that this channel wants to communicate with. 0: OUT 1: IN
14:11	EPNUM[3:0]	Endpoint number The number of the endpoint that this channel wants to communicate with.
10:0	MPL	Maximum packet length The target endpoint's maximum packet length.

Host channel-x interrupt flag register (USBHS_HCHxINTF) (x = 0..11, 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 gets 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)



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 USBHS_HCHxLEN register.
9	REQOVR	Request queue overrun The periodic request queue is full when software starts new transfer.

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 to the endpoint's maximum packet length.
7	USBER	USB Bus Error The USB error flag is set when the following conditions occur 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	NYET	NYET A NYET response packet received (in High-Speed).
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	DMAER	DMA Error An error occurs when DMA tries to fetch or write packet data for this channel.
1	CH	Channel halted When DMA is not enabled: This channel is disabled by the software request. When DMA is enabled: This channel is disabled by DMA because all the transactions of this channel finish successfully or an USB error occurs.
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 bit in USBHS_HCHxLEN register reaches to zero. For OUT channel, this flag will be triggered when software reads and pops a TF status entry from the RxFIFO.

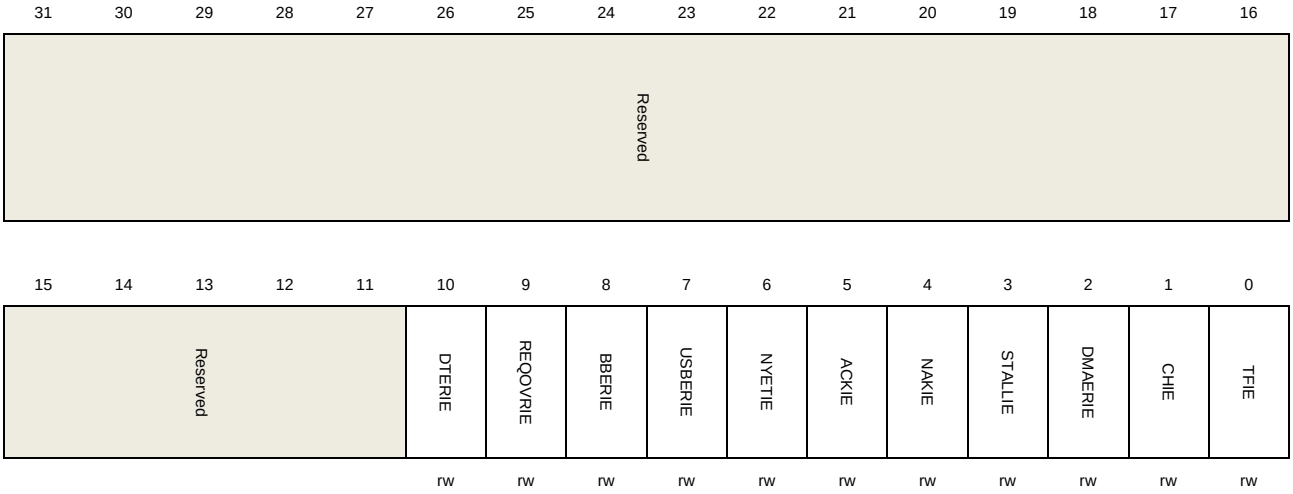
Host channel-x interrupt enable register (USBHS_HCHxINTEN) (x = 0..11, where x = channel number)

Address offset: 0x050C + (channel_number × 0x20)
Reset value: 0x0000 0000

This register contains the interrupt enabled bits for the flags in USBHS_HCHxINTF register. If a bit in this register is set by software, the corresponding bit in USBHS_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)



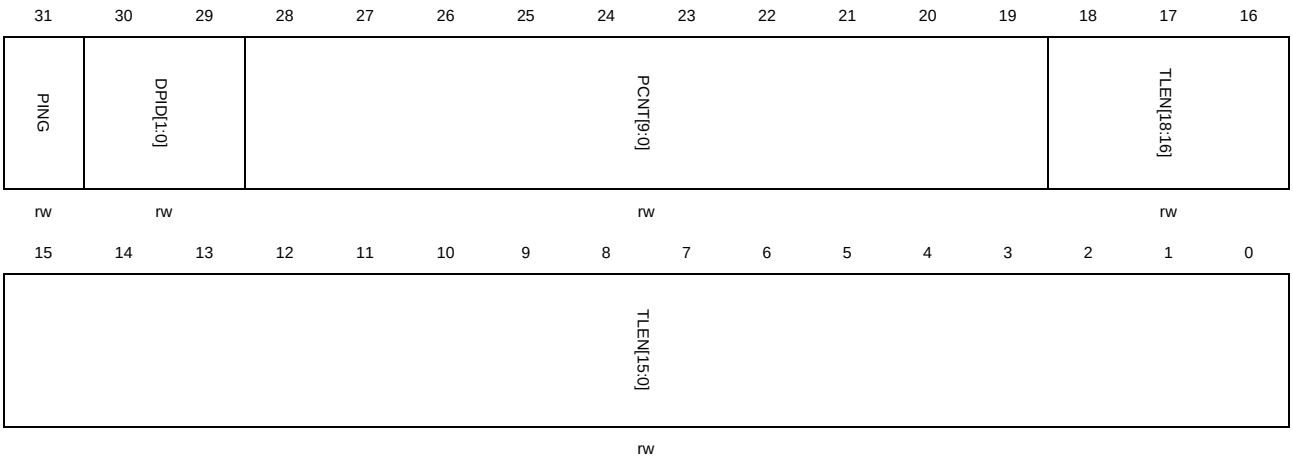
Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value.
10	DTERIE	Data toggle error interrupt enable 0: Disable data toggle error interrupt 1: Enable data toggle error interrupt
9	REQOVRIE	Request queue overrun interrupt enable 0: Disable request queue overrun interrupt 1: Enable request 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	NYETIE	NYET interrupt enable 0: Disable NYET interrupt 1: Enable NYET interrupt
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	DMAERIE	DMA Error interrupt enable 0: Disable DMA Error interrupt 1: Enable DMA Error interrupt
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 (USBHS_HCHxLEN) (x = 0..11, where x = channel number)

Address offset: 0x0510 + (channel_number × 0x20)
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31	PING	PING token request For OUT transfer, USBHS will perform PING protocol if software sets this bit. USBHS will automatically set this bit when an OUT transaction receives a NAK or NYET handshake. Do not set this bit for IN transfer.
30:29	DPID[1:0]	Data PID Software should write this field before the transfer starts. For OUT transfer, this field

controls the Data PID of the first transmitted packet. For IN transfer, 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, USBHS changes and toggles this field automatically following the USB protocol.
00: DATA0
01: DATA2
10: DATA1
11: MDATA (non-control)/SETUP (control)

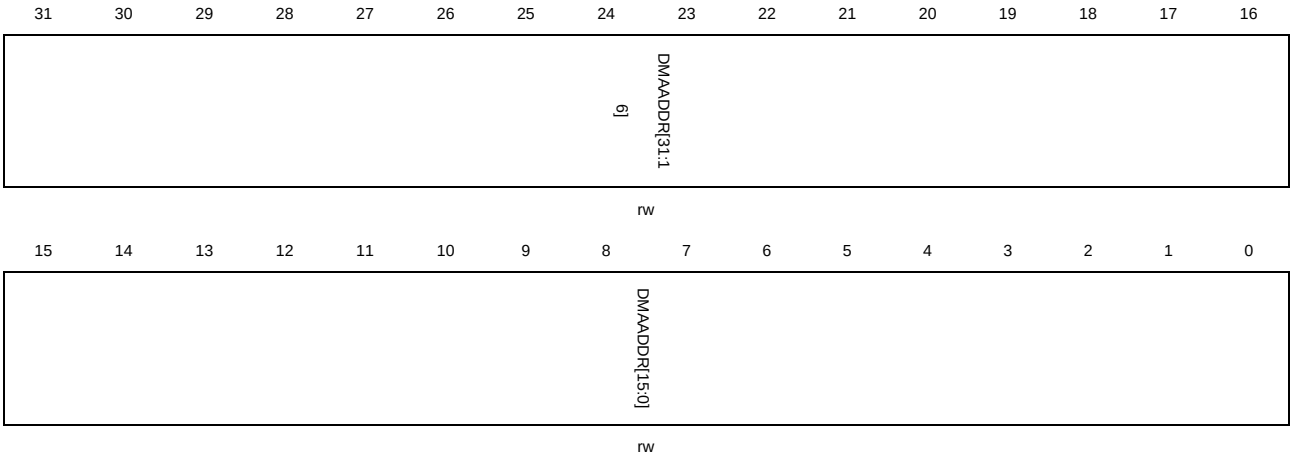
28:19 PCNT[9:0] Packet count
The number of data packets desired to be transmitted (OUT) or received (IN) in transfer. Software should program this field before the channel is enabled. After the transfer starts, this field is decreased automatically by USBHS after each successful data packet transmission.

18:0 TLEN[18:0] Transfer length
The total data bytes number of a transfer.
For OUT transfer, 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 or DMA 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.

Host channel-x DMA address register (USBHS_HCHxDMAADDR) (x = 0..11, where x = channel number)

Address offset: 0x0514 + (channel_number × 0x20)
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions

31:0	DMAADDR[31:0]	DMA address This field defines the endpoint's DMA address. DMA uses this address to fetch or write packet data for this channel.

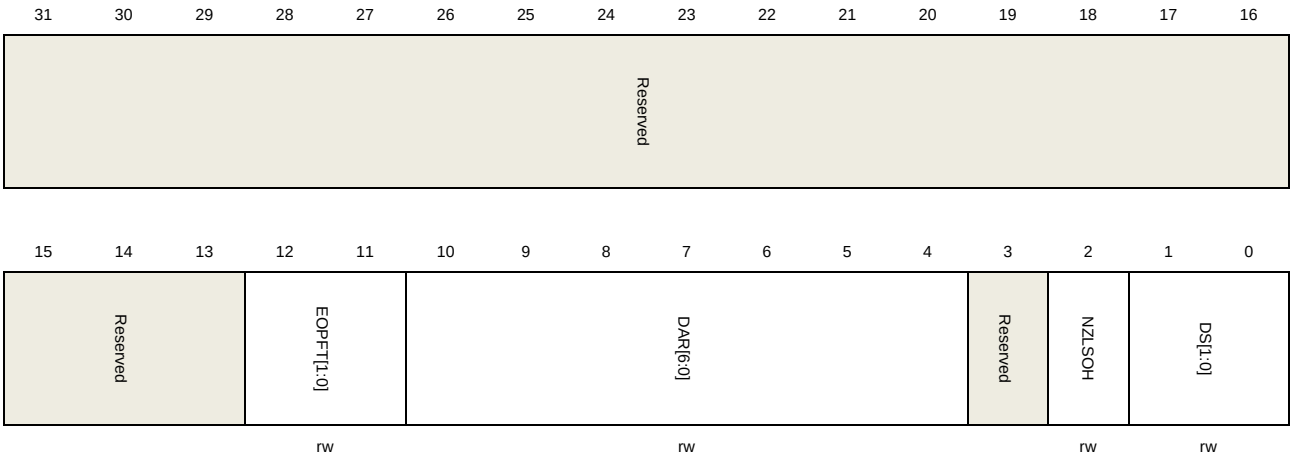
29.7.3. Device control and status registers

Device configuration register (USBHS_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)



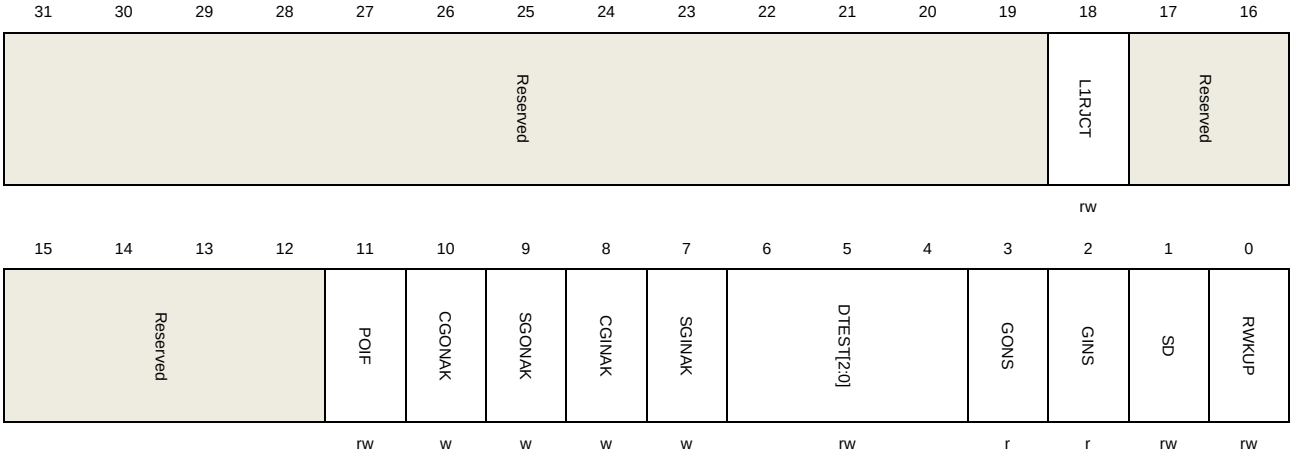
Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12:11	EOPFT[1:0]	End of periodic frame time This field defines the percentage time point in a frame when the end of periodic frame (EOPF) flag should be triggered. 00: 80% of the frame time 01: 85% of the frame time 10: 90% of the frame time 11: 95% of the frame time
10:4	DAR[6:0]	Device address This field defines the USB device's address. USBHS 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 USBHS 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 USBHS_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 is connected to a host. 00: High-Speed 01: Full-Speed Others: Reserved

Device control register (USBHS_DCTL)

Address offset: 0x0804
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value.
18	L1RJCT	Deep sleep reject When this bit is set, the core response NYET for LPM transaction with BESL greater than BSELTH.
17:12	Reserved	Must be kept at reset value.
11	POIF	Power-on initialization finished Software should set this bit to notify USBHS that the registers are initialized after waking up from power off 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 USBHS_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 USBHS_GINTF register triggered after a while. Software should clear the GINAK flag before writing this bit again.
6:4	DTEST[2:0]	Device Test control The software writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is sended on the port. When test mode is used, the HS_CUR_FE bit in USBHS_GUSBCS register should also be set. 000: Test mode disabled 001: Test_J mode 010: Test_K mode 011: Test_SE0_NAK mode 100: Test_Packet mode 101: Test_Force_Enable Others: Reserved
3	GONS	Global OUT NAK status 0: The handshake that USBHS 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: USHBS 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, USBHS first falls back to Full-Speed if currently operating at High-Speed, and then switches off the pull up resistor on DP line. This will cause the host to detect a device disconnect. 0: No soft disconnect generated.

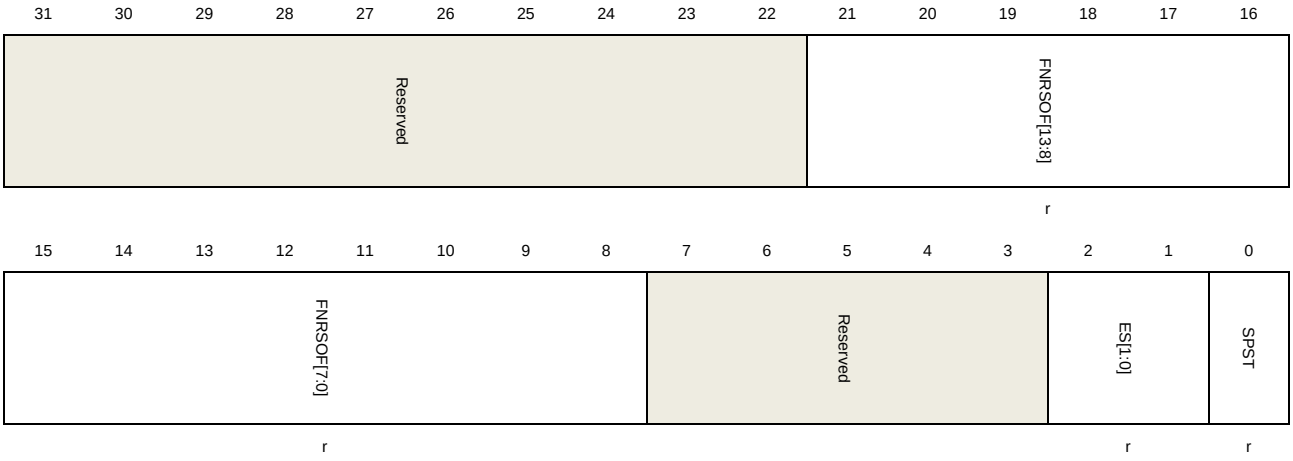
		1: Generate a soft disconnect.
0	RWKUP	Remote wakeup In suspend state, software can use this bit to generate a Remote wake up signal to inform host that it should resume the USB bus. 0: No remote wakeup signal generated. 1: Generate remote wakeup signal. When the core is LPM enabled and in sleep status, if this bit is set, the core continues to drive it and clears it automatically after 50us (TL1DevDrvResume). The application cannot set this bit when bRemoteWake value received from LPM transaction is zero.

Device status register (USBHS_DSTAT)

Address offset: 0x0808
Reset value: 0x0000 0000

This register contains status and information of the USBHS in device mode.

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



Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value.
21:8	FNRSOF[13:0]	The frame number of the received SOF. USBHS 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. Software should read this field after the ENUMF flag in USBHS_GINTF register is triggered. 00: High-Speed 01: Full-Speed

Others: reserved

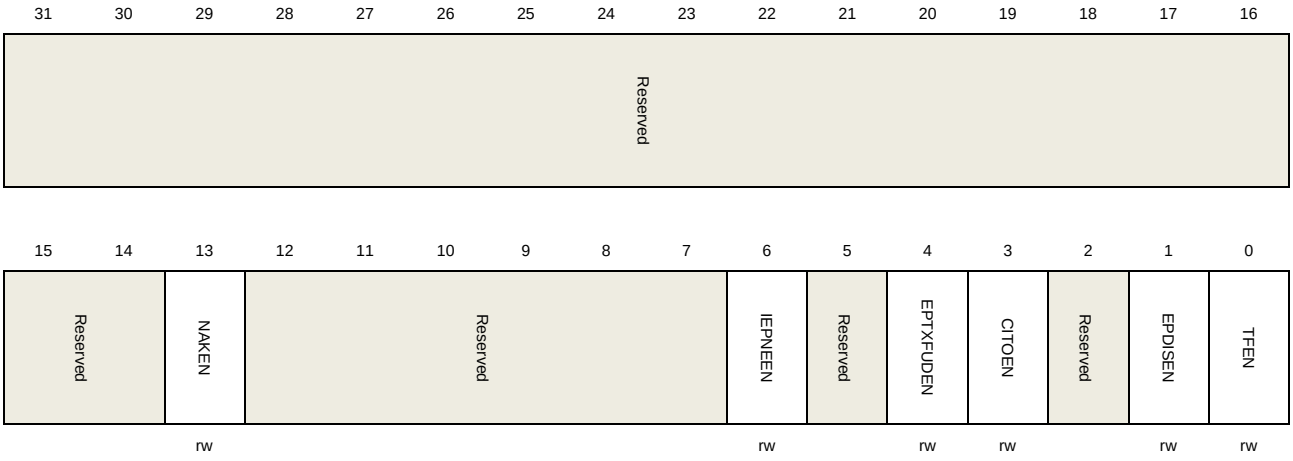
0	SPST	Suspend status This bit reports whether device is in suspend state. 0: Device is in suspend state. 1: Device is not in suspend state.

Device IN endpoint common interrupt enable register (USBHS_DIEPINTEN)

Address offset: 0x810
Reset value: 0x0000 0000

This register contains the interrupt enabled bits for the flags in USBHS_DIEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBHS_DIEPxINTF register is able to trigger an endpoint interrupt in USBHS_DAEPINT register. The bits in this register are set and cleared by software.

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



Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13	NAKEN	NAK handshake sent by USBHS interrupt enable bit 0: Disable interrupt 1: Enable interrupt
12:7	Reserved	Must be kept at reset value.
6	IEPNEEN	IN endpoint NAK effective interrupt enable bit 0: Disable interrupt 1: Enable interrupt
5	Reserved	Must be kept at reset value.
4	EPTXFUDEN	Endpoint Tx FIFO underrun interrupt enable bit 0: Disable interrupt

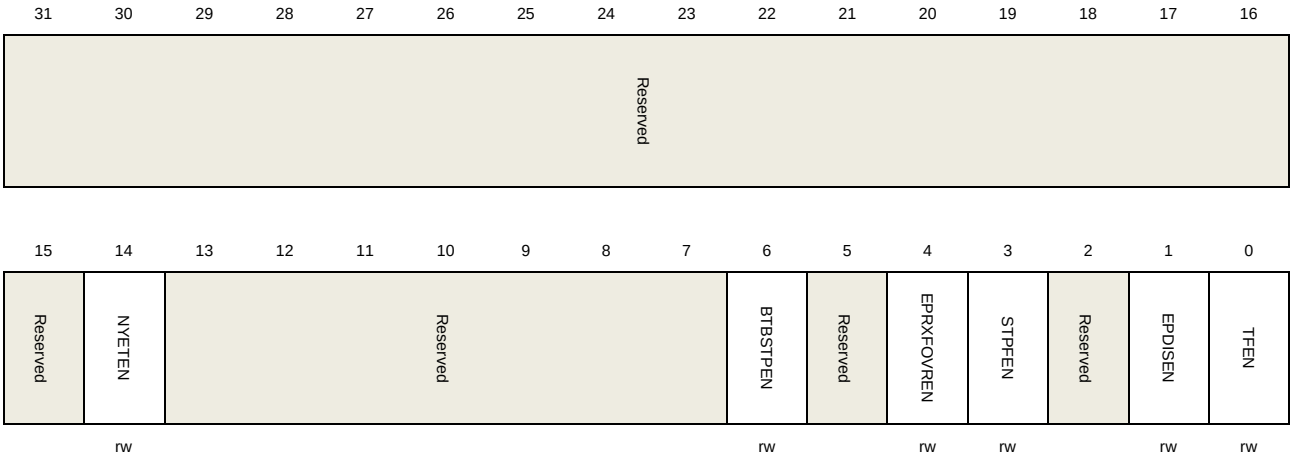
		1: Enable interrupt
3	CITOEN	Control In Timeout interrupt enable bit 0: Disable interrupt 1: Enable interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable interrupt 1: Enable interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable interrupt 1: Enable interrupt

Device OUT endpoint common interrupt enable register (USBHS_DOEPINTEN)

Address offset: 0x0814
Reset value: 0x0000 0000

This register contains the interrupt enabled bits for the flags in USBHS_DOEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBHS_DOEPxINTF register is able to trigger an endpoint interrupt in USBHS_DAEPINT register. The bits in this register are set and cleared by software.

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



Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	NYETEN	Send NYET handshake interrupt enable bit 0: Disable interrupt 1: Enable interrupt

13:7	Reserved	Must be kept at reset value.
6	BTBSTPEN	Back-to-back SETUP packets (Only for control OUT endpoint) interrupt enable bit 0: Disable interrupt 1: Enable interrupt
5	Reserved	Must be kept at reset value.
4	EPRXFOVREN	Endpoint Rx FIFO overrun interrupt enable bit 0: Disable interrupt 1: Enable interrupt
3	STPFEN	SETUP phase finished (Only for control OUT endpoint) interrupt enable bit 0: Disable interrupt 1: Enable interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable interrupt 1: Enable interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable interrupt 1: Enable interrupt

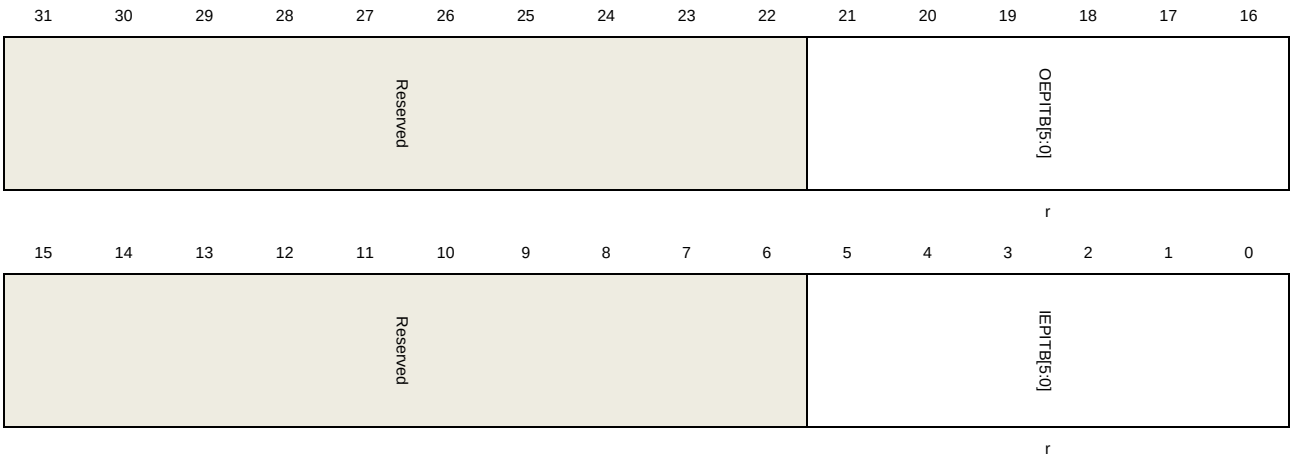
Device all endpoints interrupt register (USBHS_DAEPINT)

Address offset: 0x0818

Reset value: 0x0000 0000

When an endpoint interrupt is triggered, USBHS 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)



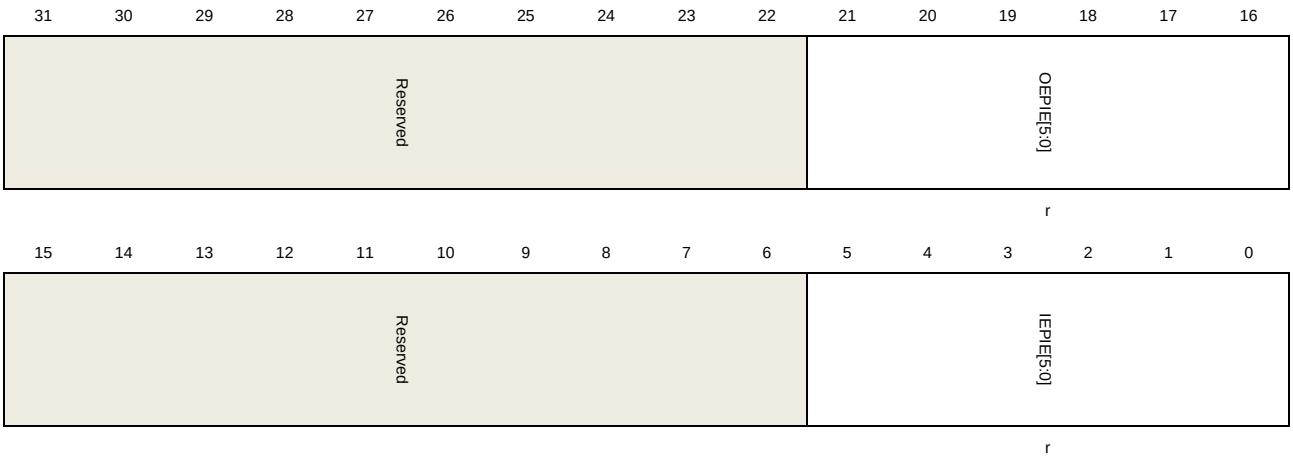
Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value.
21:16	OEPITB[5:0]	Device all OUT endpoints interrupt bits Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 21 for OUT endpoint 5.
15:6	Reserved	Must be kept at reset value.
5:0	IEPITB[5:0]	Device all IN endpoints interrupt bits Each bit represents an IN endpoint: Bit 0 for IN endpoint 0, bit 5 for IN endpoint 5.

Device all endpoints interrupt enable register (USBHS_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 USBHS_GINTF register.

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



Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value.
21:16	OEPIE[5:0]	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 21 for OUT endpoint 5.
15:6	Reserved	Must be kept at reset value.
5:0	IEPIE[5:0]	IN endpoint interrupt enable bits

0: Disable IN endpoint-n interrupt

1: Enable IN endpoint-n interrupt

Each bit represents an IN endpoint:

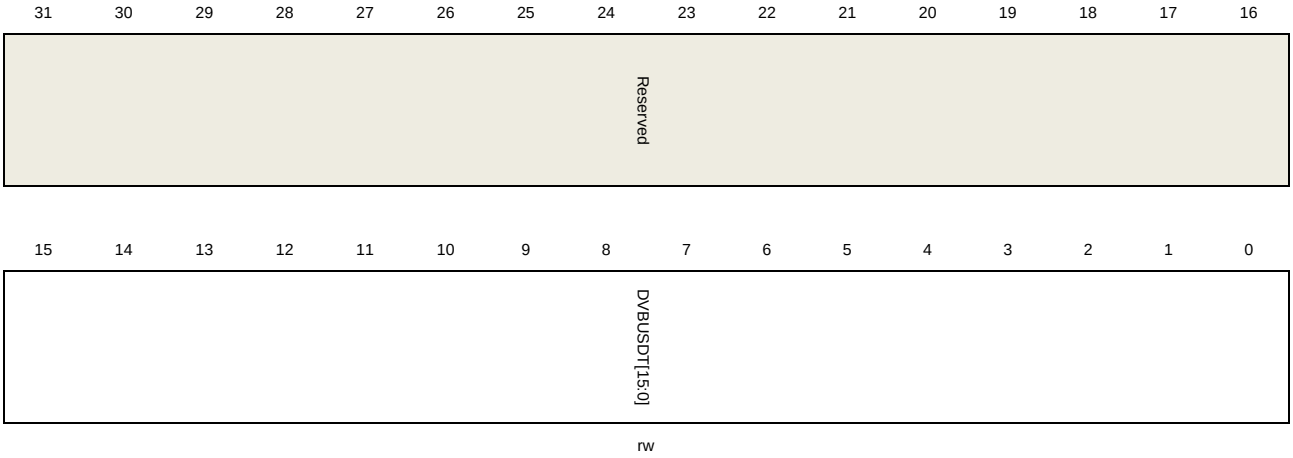
Bit 0 for IN endpoint 0, bit 5 for IN endpoint 5.

Device VBUS discharge time register (USBHS_DVBUSDT)

Address offset: 0x0828

Reset value: 0x0000 17D7

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



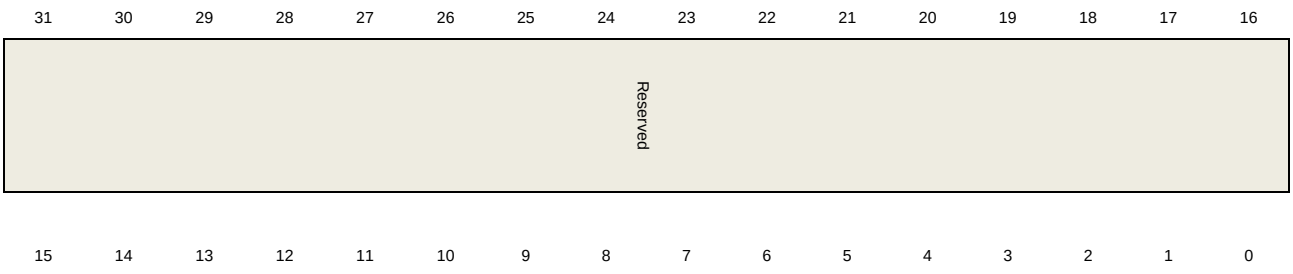
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	DVBUSDT[15:0]	Device V_{BUS} discharge time There is a discharge process after V_{BUS} pulsing in SRP protocol. This field defines the discharge time of V_{BUS}. The true discharge time is 1024*DVBUSDT[15:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ is the period time of USB clock.

Device VBUS pulsing time register (USBHS_DVBUSPT)

Address offset: 0x082C

Reset value: 0x0000 05B8

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



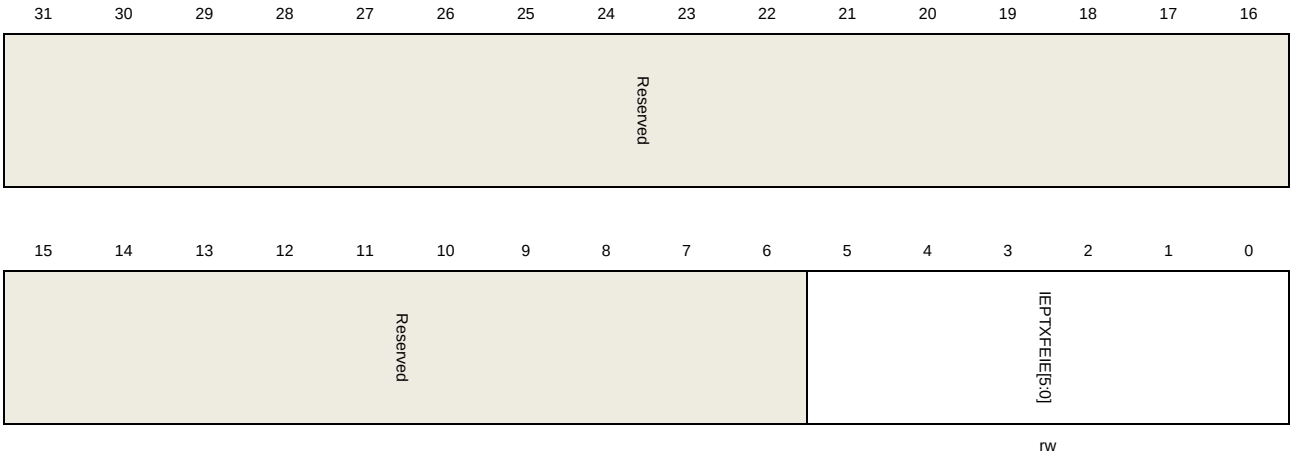
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	DVBUSPT[11:0]	Device V_{BUS} pulsing time This field defines the pulsing time for V_{BUS}. The true pulsing time is 1024*DVBUSPT[15:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ is the period time of USB clock.

Device IN endpoint FIFO empty interrupt enable register (USBHS_DIEPFEINTEN)

Address offset: 0x0834
Reset value: 0x0000 0000

This register contains the enabled bits for the Tx FIFO empty interrupts of IN endpoints.

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



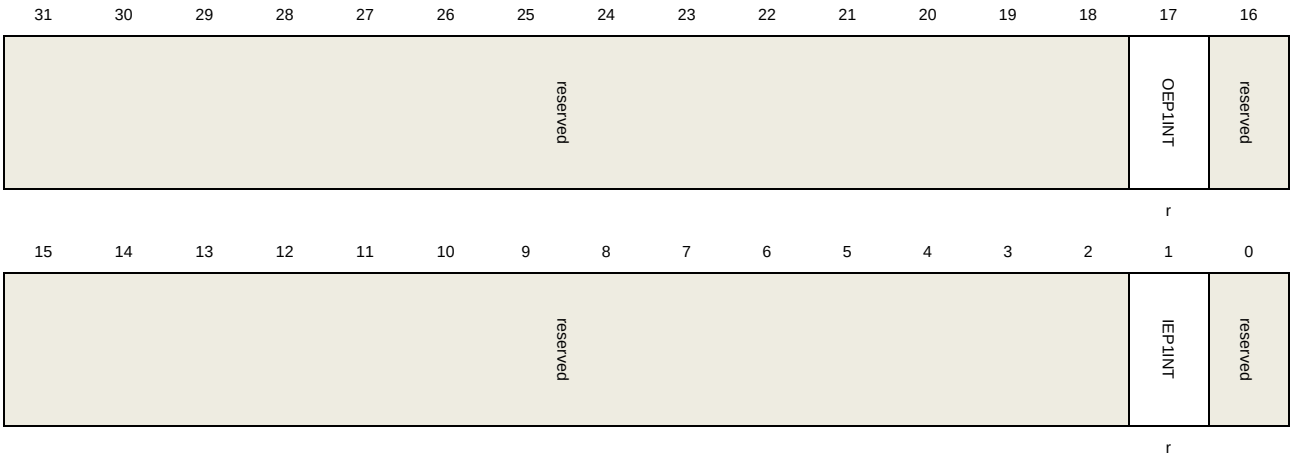
Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5:0	IEPTXFEIE[5:0]	IN endpoint Tx FIFO empty interrupt enable bits This field controls whether the TXFE bit in USBHS_DIEPxINTF register is able to generate an endpoint interrupt bit in USBHS_DAEPINT register. Bit 0 for IN endpoint 0, bit 5 for IN endpoint 5 0: Disable FIFO empty interrupt 1: Enable FIFO empty interrupt

Device endpoint 1 interrupt register (USBHS_DEP1INT)

Address offset: 0x0838
Reset value: 0x0000 0000

When ep1 out or in interrupt is triggered, USBHS sets corresponding bit in this register and software should read this register to know which endpoint is asserting the ep1 interrupt.

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



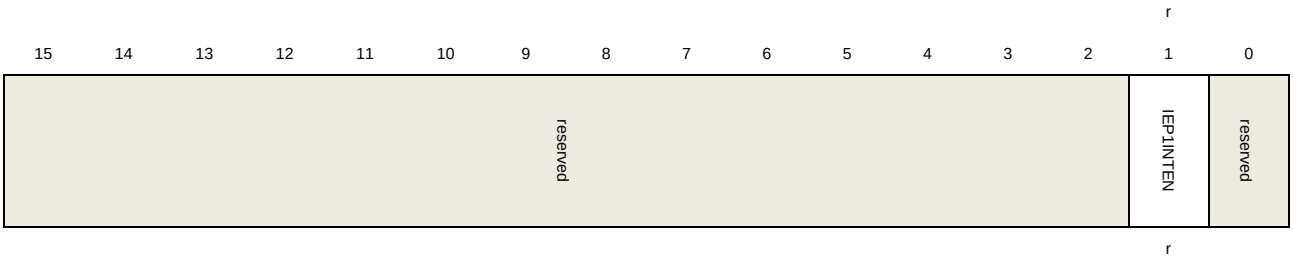
Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.
17	OEP1INT	OUT Endpoint 1 interrupt
16:2	Reserved	Must be kept at reset value.
1	IEP1INT	IN Endpoint 1 interrupt
0	Reserved	Must be kept at reset value.

Device endpoint 1 interrupt enable register (USBHS_DEP1INTEN)

Address offset: 0x083C
Reset value: 0x0000 0000

This register can be used by software to enable or disable endpoint-1's interrupt. Only the endpoint whose corresponding bit in this register is set is able to cause the endpoint-1 in or out interrupt.

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



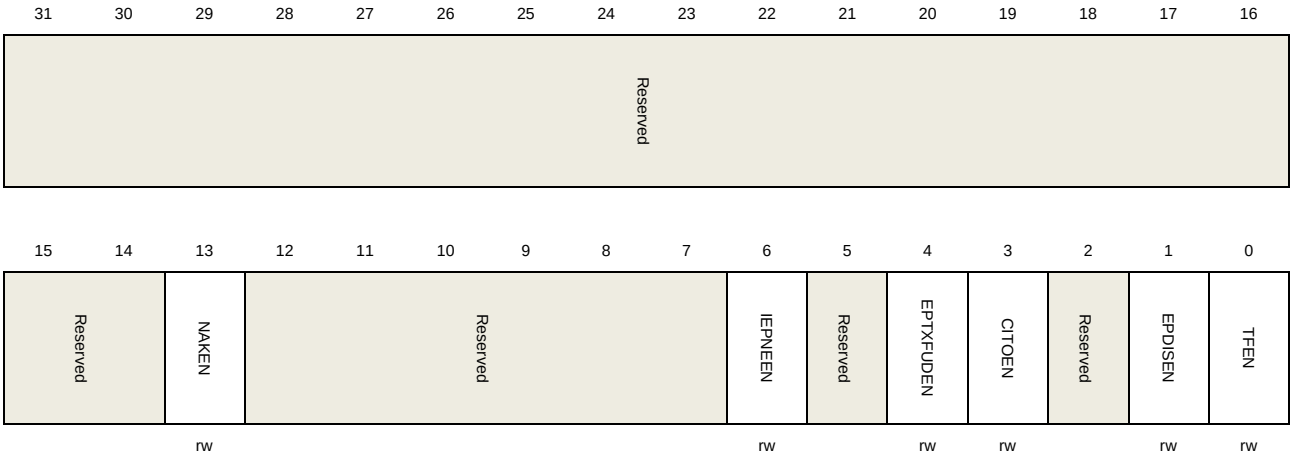
Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.
17	OEP1INTEN	OUT Endpoint 1 interrupt enable
16:2	Reserved	Must be kept at reset value.
1	IEP1INTEN	IN Endpoint 1 interrupt enable
0	Reserved	Must be kept at reset value.

Device IN endpoint-1 interrupt enable register (USBHS_DIEP1INTEN)

Address offset: 0x844
Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBHS_DIEP1INTF register. If a bit in this register is set by software, the corresponding bit in USBHS_DIEP1INTF register is able to trigger an endpoint interrupt in USBHS_DEP1INT register. The bits in this register are set and cleared by software.

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



Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13	NAKEN	Interrupt enable bit of NAK handshake sent by USBHS 0: Disable interrupt

		1: Enable interrupt
12:7	Reserved	Must be kept at reset value.
6	IEPNEEN	IN endpoint NAK effective interrupt enable bit 0: Disable interrupt 1: Enable interrupt
5	Reserved	Must be kept at reset value.
4	EPTXFUDEN	Endpoint Tx FIFO underrun interrupt enable bit 0: Disable interrupt 1: Enable interrupt
3	CITOEN	Control In Timeout interrupt enable bit 0: Disable interrupt 1: Enable interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable interrupt 1: Enable interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable interrupt 1: Enable interrupt

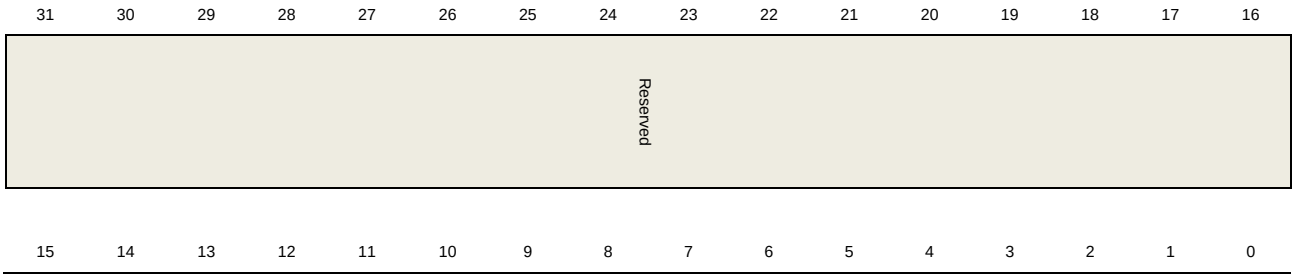
Device OUT endpoint-1 interrupt enable register (USBHS_DOEP1INTEN)

Address offset: 0x0884

Reset value: 0x0000 0000

This register contains the interrupt enabled bits for the flags in USBHS_DOEP1INTF register. If a bit in this register is set by software, the corresponding bit in USBHS_DOEP1INTF register is able to trigger an endpoint interrupt in USBHS_DEP1INT register. The bits in this register are set and cleared by software.

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



Reserved	NYETEN	Reserved	BTBSTPEN	Reserved	EPRXFOVREN	STPFEN	Reserved	EPDISEN	TFEN
	rw		rw		rw	rw		rw	rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	NYETEN	Send NYET handshake interrupt enable bit 0: Disable interrupt 1: Enable interrupt
13:7	Reserved	Must be kept at reset value.
6	BTBSTPEN	Back-to-back SETUP packets (Only for control OUT endpoint) interrupt enable bit 0: Disable interrupt 1: Enable interrupt
5	Reserved	Must be kept at reset value.
4	EPRXFOVREN	Endpoint Rx FIFO overrun interrupt enable bit 0: Disable interrupt 1: Enable interrupt
3	STPFEN	SETUP phase finished (Only for control OUT endpoint) interrupt enable bit 0: Disable interrupt 1: Enable interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable interrupt 1: Enable interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable interrupt 1: Enable interrupt

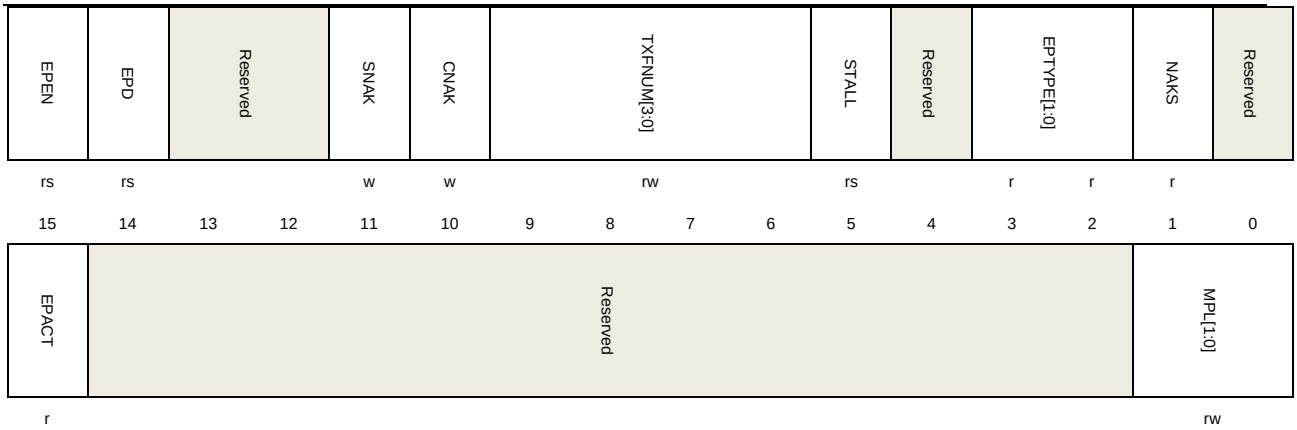
Device IN endpoint 0 control register (USBHS_DIEP0CTL)

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



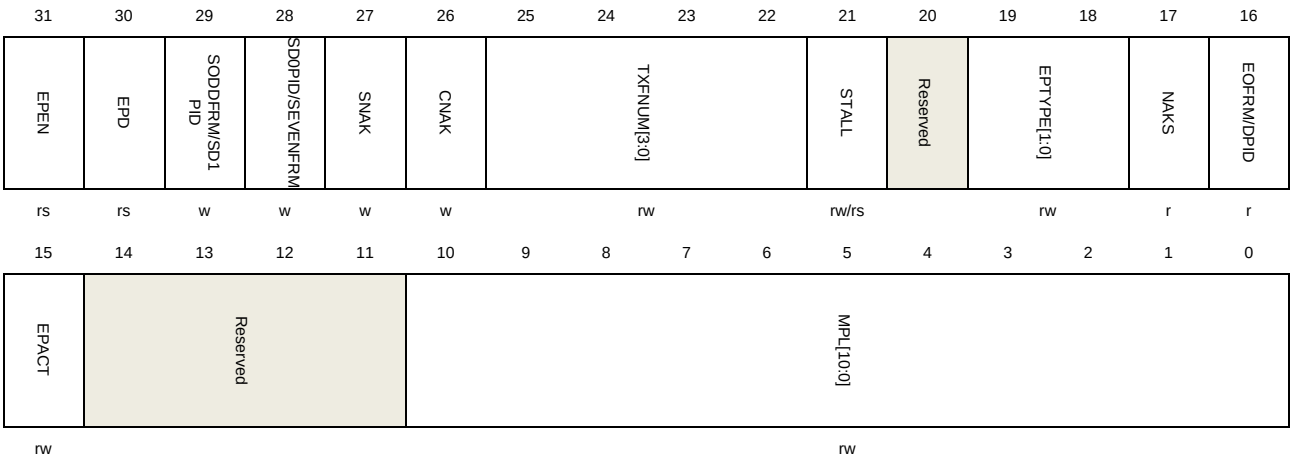
Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBHS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable Software can set this bit to disable the endpoint. Software should follow the operation guide to disable or enable an endpoint.
29:28	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 Define the Tx FIFO number of IN endpoint 0.
21	STALL	STALL handshake Software can set this bit to make USBHS send STALL handshake when receiving IN token. USBHS 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 USBHS_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 USBHS when both STALL bit in this register and GINS bit in USBHS_DCTL register are cleared:

		0: USBHS sends data or handshake packets according to the status of the endpoint's Tx FIFO. 1: USBHS 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 (USBHS_DIEPxCTL) (x = 1..5, where x = endpoint_number)

Address offset: 0x0900 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBHS. 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 USBHS send STALL handshake when receiving IN token. This bit has a higher priority than NAKS bit in this register and GINS bit in USBHS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect. For control IN endpoint: Only USBHS 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 USBHS when both STALL bit in this register and

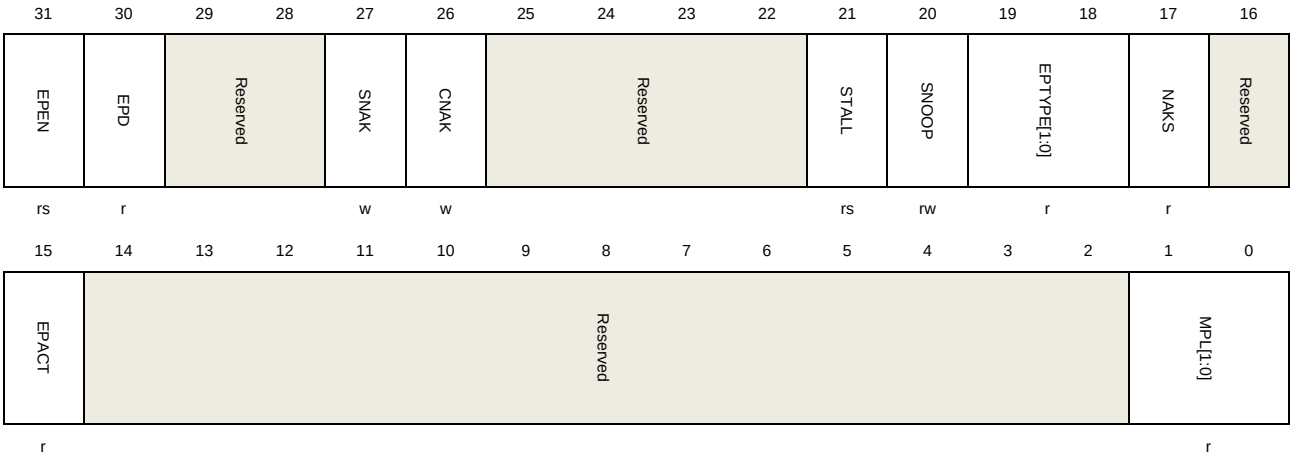
		GINS bit in USBHS_DCTL register are cleared: 0: USBHS sends data or handshake packets according to the status of the endpoint's Tx FIFO. 1: USBHS always sends NAK handshake to the IN token. This bit is read-only and software should use CNAK and SNAK in this register to control this bit.
16	EOFRM	Even/odd frame (For isochronous IN endpoints) For isochronous transfer, software can use this bit to control that USBHS only sends data packets for IN tokens in even or odd frames. If the current frame number's parity doesn't match with this bit, USBHS 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) 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 USBHS maintains this bit during transfers by 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 OUT endpoint 0 control register (USBHS_DOEP0CTL)

Address offset: 0x0B00

Reset value: 0x0000 8000

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



Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBHS. 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 Software can set this bit to make USBHS send STALL handshake during an OUT transaction. USBHS 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, USBHS 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 USBHS when both STALL bit in this register and GONS bit in USBHS_DCTL register are cleared: 0: USBHS sends data or handshake packets according to the status of the endpoint's Rx FIFO. 1: USBHS 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	Reserved	Must be kept at reset value.
15	EPACT	Endpoint active This field is fixed to '1' for endpoint 0.

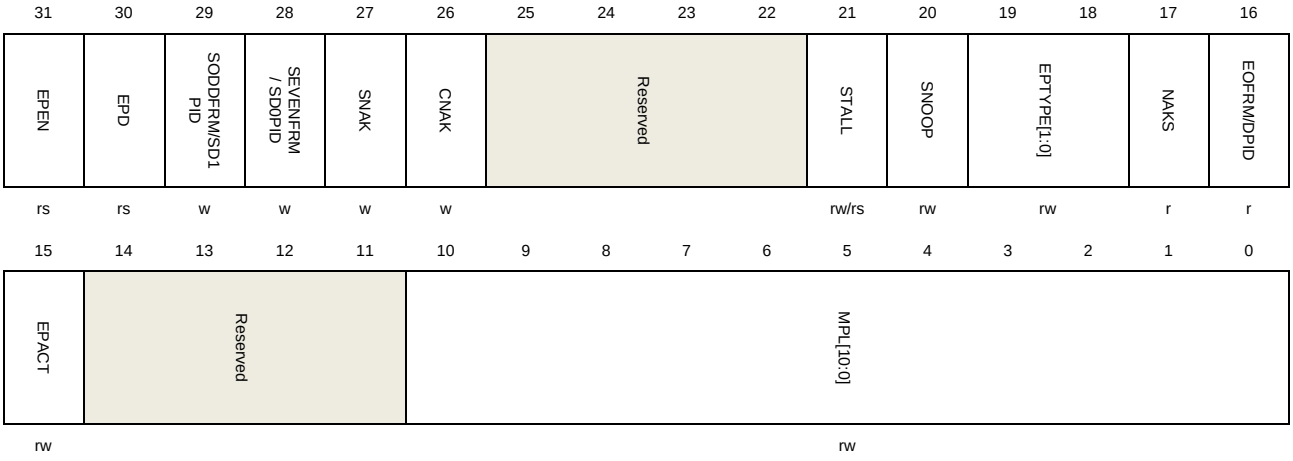
14:2	Reserved	Must be kept at reset value.
1:0	MPL[1:0]	Maximum packet length This is a read-only field, and its value comes from the MPL field of USBHS_DIEP0CTL register: 00: 64 bytes 01: 32 bytes 10: 16 bytes 11: 8 bytes

Device OUT endpoint-x control register (USBHS_DOEPxCTL) (x = 1..5, where x = endpoint_number)

Address offset: 0x0B00 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

The application uses this register to control the operation of each logical OUT endpoint other than OUT endpoint 0.

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



Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBHS. 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 USBHS send STALL handshake during an OUT transaction. This bit has a higher priority than NAKS bit in this register and GINS in USBHS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect. For control OUT endpoint: Only USBHS 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, USBHS 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 USBHS when both STALL bit in this register and GONS bit in USBHS_DCTL register are cleared: 0: USBHS sends handshake packets according to the status of the endpoint's Rx FIFO. 1: USBHS always sends NAK handshake to the OUT token.

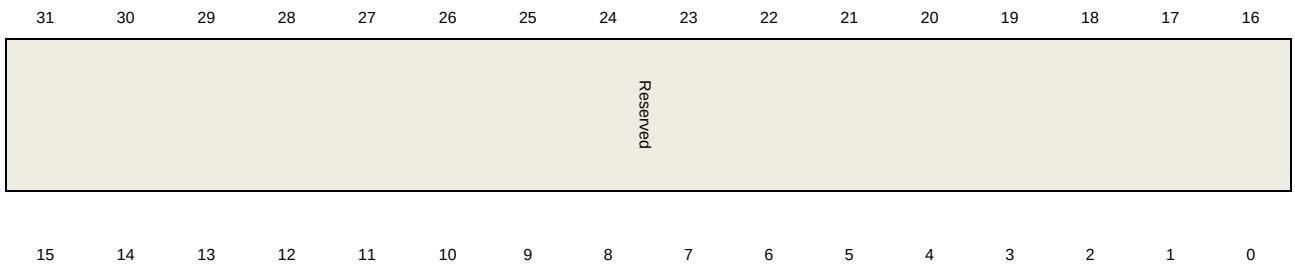
		This bit is read-only and software should use CNAK and SNAK in this register to control this bit.
16	EOFRM	Even/odd frame (For isochronous OUT endpoints) For isochronous transfer, software can use this bit to control that USBHS only receives data packets in even or odd frames. If the current frame number's parity doesn't match with this bit, USBHS 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 USBHS maintains this bit during transfer 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 (USBHS_DIEPxINTF) (x = 0..5, 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 software gets an IN endpoint interrupt, it should 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)



Reserved	NAK	Reserved	TXFE	IEPNE	Reserved	EPTXFUD	CITO	Reserved	EPDIS	TF
	rc_w1		r	rc_w1		rc_w1	rc_w1		rc_w1	rc_w1

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13	NAK	NAK handshake sent by USBHS USBHS sets this bit after it sends out a NAK handshake because the NAKS bit in USBHS_DIEPxCTL register is set, or there is no packet data in endpoint's Tx FIFO.
12:8	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 USBHS_GAHBCS register.
6	IEPNE	IN endpoint NAK effective The setting of SNAK bit in USBHS_DIEPxCTL register takes effect. This bit can be cleared either by writing 1 to it or by setting CNAK bit in USBHS_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 from the software's request.
0	TF	Transfer finished This flag is triggered when all the IN transactions assigned to this endpoint have finished.

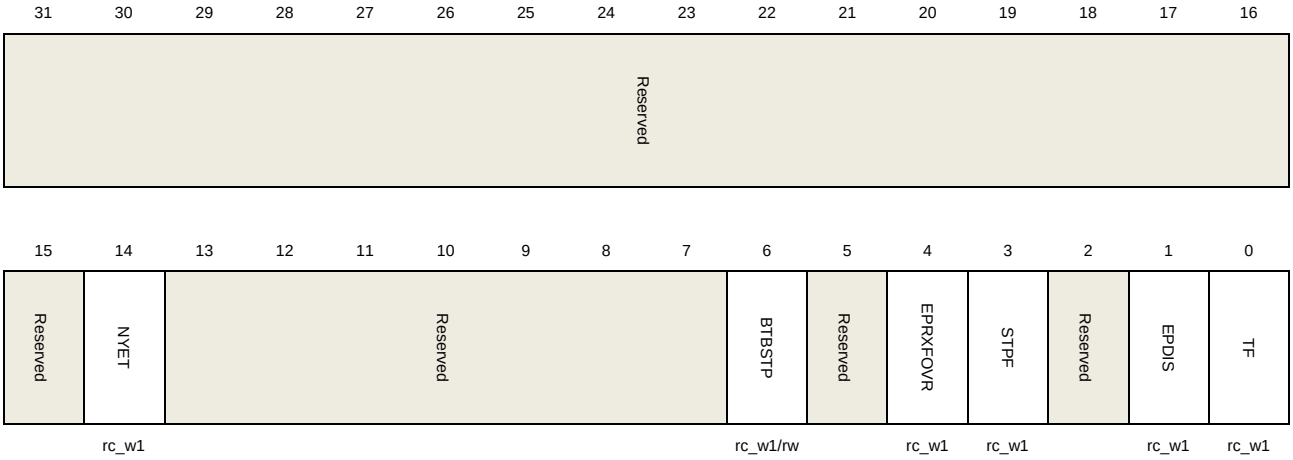
Device OUT endpoint-x interrupt flag register (USBHS_DOEPxINTF) (x = 0..5, where x = endpoint_number)

Address offset: 0x0B08 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

This register contains the status and events of an OUT endpoint, when software gets an OUT endpoint interrupt, it should 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)



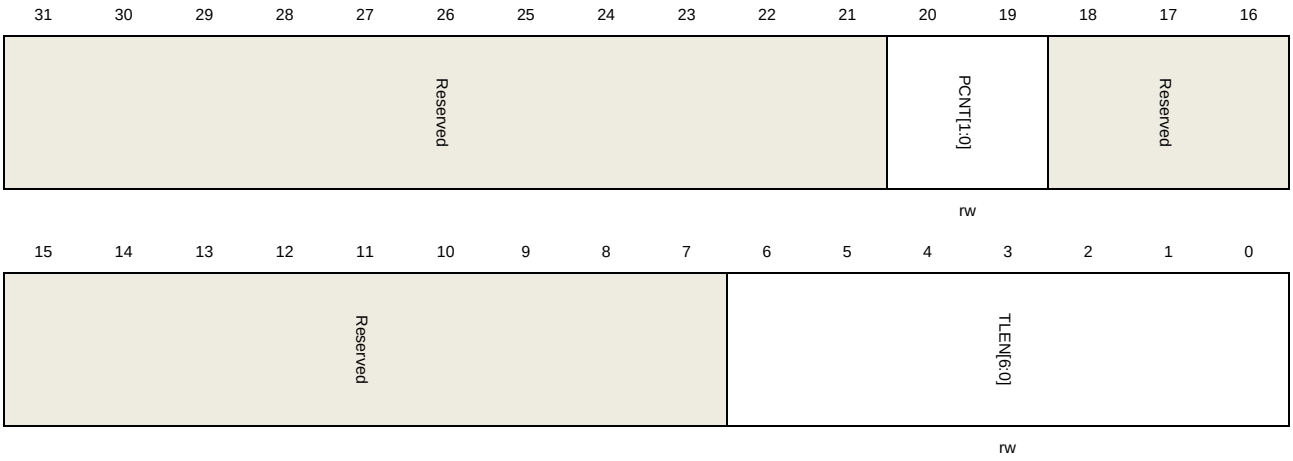
Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	NYET	NYET handshake is sent This flag is triggered if a NYET handshake is sent by USBHS.
13:7	Reserved	Must be kept at reset value.
6	BTBSTP	Back-to-back SETUP packets (Only for control OUT endpoint) This flag is triggered when a 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. USBHS will drop the incoming OUT data packet and send 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. USBHS 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 from the software's request.
0	TF	Transfer finished This flag is triggered when all the OUT transactions assigned to this endpoint have finished.

Device IN endpoint 0 transfer length register (USBHS_DIEP0LEN)

Address offset: 0x0910

Reset value: 0x0000 0000

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



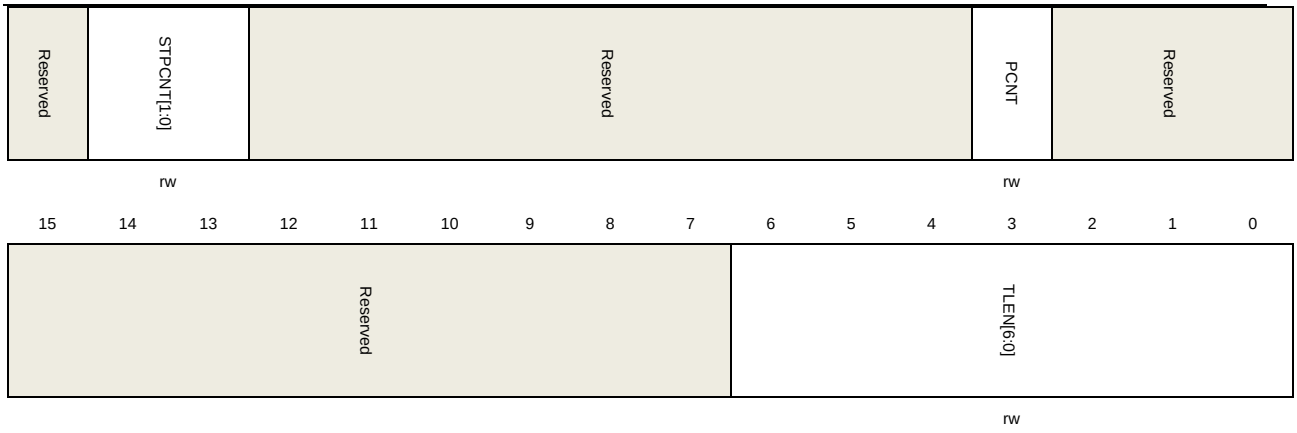
Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20:19	PCNT[1:0]	Packet count The number of data packets desired to be transmitted in a transfer. Software should program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBHS 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. Software should program this field before the endpoint is enabled. When software or DMA 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 (USBHS_DOEP0LEN)

Address offset: 0x0B10

Reset value: 0x0000 0000

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



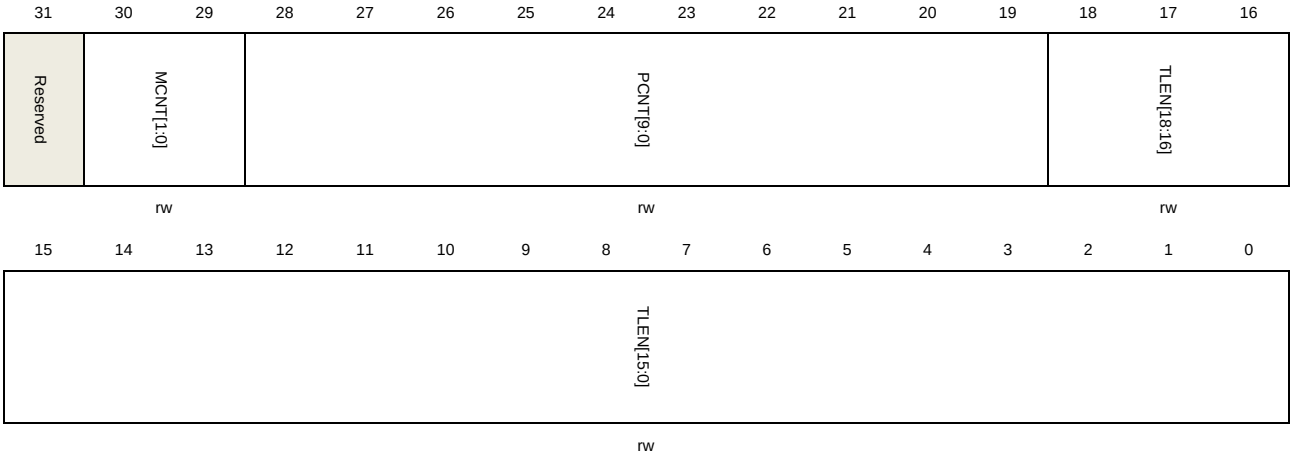
Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30:29	STPCNT[1:0]	SETUP packet count This field defines the maximum number of back-to-back SETUP packets what this endpoint can accept. Software should program this field before setup transfers. Each time a back-to-back setup packet is received, USBHS decreases this field by one. When this field reaches zero, the BTBSTP flag in USBHS_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 is desired to receive in a transfer. Software should program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBHS 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. Software should program this field before the endpoint is enabled. Each time software or DMA 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 (USBHS_DIEPxLEN) (x = 1..5, where x = endpoint_number)

Address offset: 0x910 + (endpoint_number × 0x20)

Reset value: 0x0000 0000

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

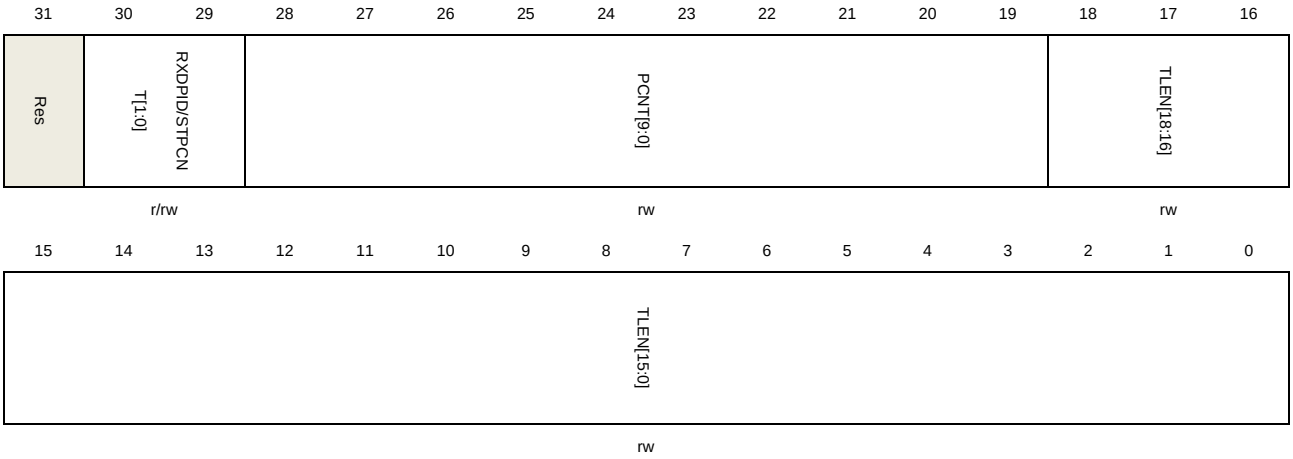


Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30:29	MCNT[1:0]	Multi count This field indicates the number of packets which should be transmitted in a frame 01: 1 packet 10: 2 packets 11: 3 packets
28:19	PCNT[9:0]	Packet count The number of data packets desired to be transmitted in a transfer. Software should program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBHS 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. Software should program this field before the endpoint is enabled. When software or DMA 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 (USBHS_DOEPxLEN) (x = 1..5, where x = endpoint_number)

Address offset: 0x0B10 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

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

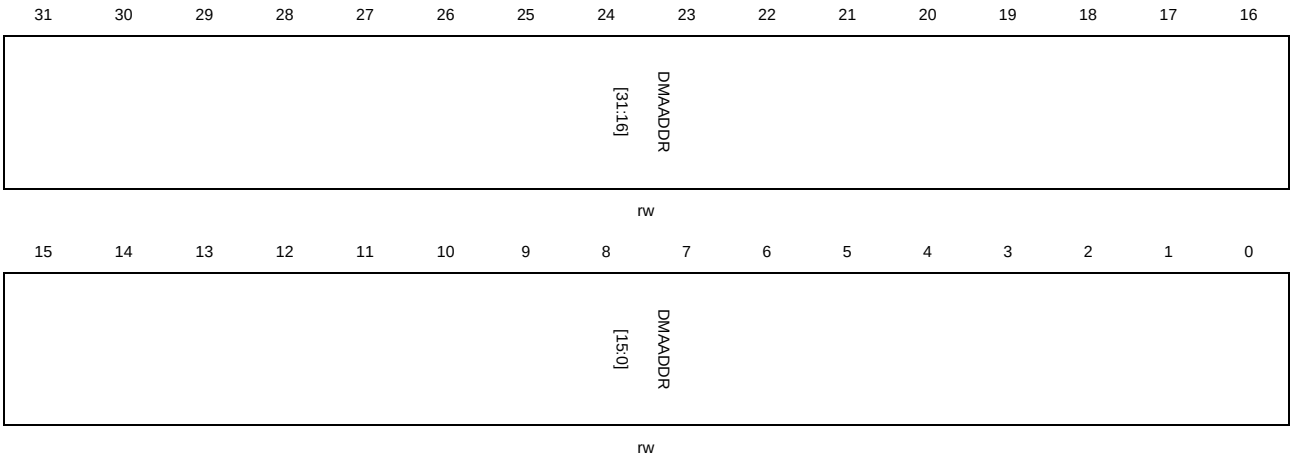


Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30:29	RXDPID[1:0]	Received data PID (For isochronous OUT endpoints) This field saves the PID of the latest received data packet on this endpoint. 00: DATA0 01: DATA2 10: DATA1 11: MDATA
	STPCNT[1:0]	SETUP packet count (For control OUT Endpoints.) This field defines the maximum number back-to-back SETUP packets this endpoint can accept. Software should program this field before setup transfers. Each time a back-to-back setup packet is received, USBHS decreases this field by one. When this field reaches zero, the BTBSTP flag in USBHS_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. Software should program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBHS 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. Software should program this field before the endpoint is enabled. Each time software or DMA reads out a packet from the RxFIFO, this field is decreased by the byte size of the packet.

Device IN endpoint-x DMA address register (USBHS_DIEPxDMAADDR) / Device OUT endpoint-x DMA address register (USBHS_DOEPxDMAADDR) (x = 0..5, where x = endpoint_number)

Address offset:
IN endpoint: 0x0914 + (endpoint_number × 0x20)
OUT endpoint: 0x0B14 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

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



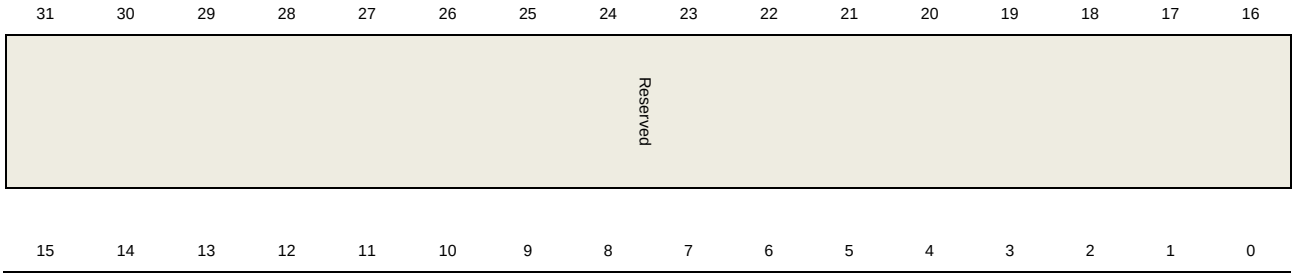
Bits	Fields	Descriptions
31:0	DMAADDR[31:0]	DMA address This field defines the endpoint's DMA address. DMA uses this address to fetch packet data for IN endpoint or write packet data for OUT endpoint.

Device IN endpoint-x transmit FIFO status register (USBHS_DIEPxTFSTAT) (x = 0..5, 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)

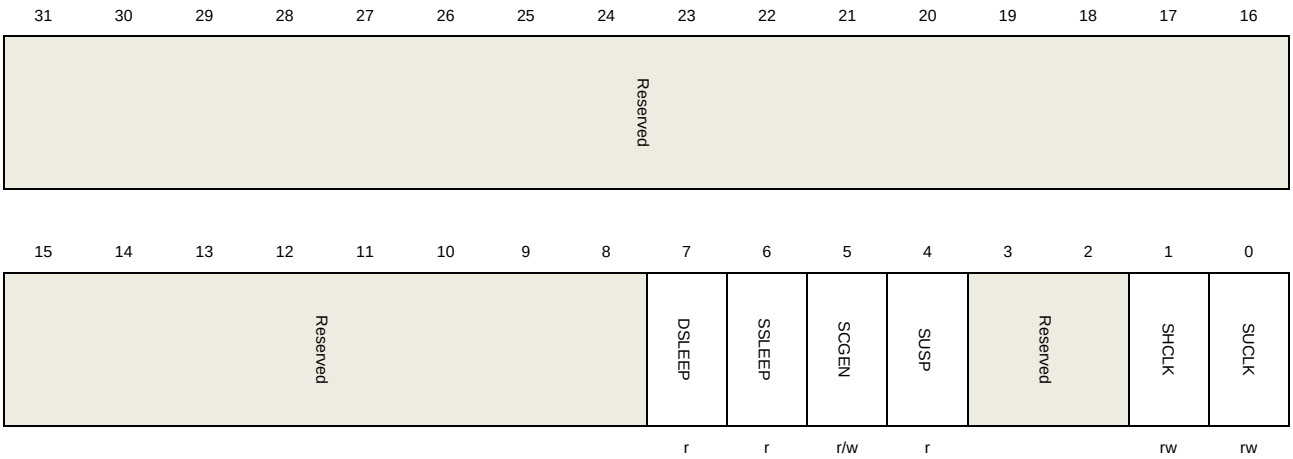


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	IEPTFS[15:0]	IN endpoint's Tx FIFO space remaining IN endpoint's Tx FIFO space remaining in 32-bit word: 0: FIFO is full 1: 1 word available … n: n words available

29.7.4. Power and clock control register (USBHS_PWRCLKCTL)

Address offset: 0x0E00
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	DSLEEP	PHY is in deep sleep status
6	SSLEEP	PHY is in shallow sleep status
5	SCGEN	When this bit is set, the internal clock gating is enabled.
4	SUSP	PHY is in suspend status
3:2	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: UCB clock is stopped

30. Appendix

30.1. List of abbreviations used in register

Table 30-1. List of abbreviations used in register

abbreviations for registers	Descriptions
read/write (rw)	Software can read and write to this bit.
read-only (r)	Software can only read this bit.
write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
read/clear write 1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
read/clear write 0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value.
toggle (t)	The software can toggle this bit by writing 1. Writing 0 has no effect.
read/set (rs)	Software can read as well as set this bit to 1. Writing '0' has no effect on the bit value.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to '0'. Writing '0' has no effect on the bit value.

30.2. List of terms

Table 30-2. List of terms

Glossary	Descriptions
Word	Data of 32-bit length.
Half-word	Data of 16-bit length.
Byte	Data of 8-bit length.
IAP (in-application programming)	Writing 0 has no effect IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running.
ICP (in-circuit programming)	ICP is the ability to program the Flash memory of a microcontroller using the JTAG protocol, the SWD protocol or the boot loader while the device is mounted on the user application board.
Option bytes	Product configuration bits stored in the Flash memory.
AHB	Advanced high-performance bus.
APB	Advanced peripheral bus.
RAZ	Read-as-zero.
WI	Writes ignored.
RAZ/WI	Read-as-zero, writes ignored.

30.3. Available peripherals

For availability of peripherals and their number across all MCU series types, refer to the corresponding device data datasheet.

31. Revision history

Table 31-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Feb.21, 2020
1.1	Module information update	Aug.28, 2020
1.2	1. Modify the voltage difference value when VDDA and VDD are different in the PMU chapter,refers to ***VDDA domain***. 2. Modify the offset address of the PID register in the FMC chapter to 0x104,refers to ***Product ID register (FMC_PID)***. 3. M33 core does not support bit-band operation, delete the original description about bit-band operation in chapter 1.3. 4. Modify the length of the pulse signal generated by the SOF in the ***USB host function*** chapter from the original 16 HCLK cycles to 12 HCLK cycles. 5. USART5 does not support hardware flow control, delete its content related to hardware flow control, refers to ***Universal synchronous/asynchronous receiver /transmitter (USARTx, x=5)***.	Dec.23, 2020
1.3	Add CAN module.	Mar.30, 2020
1.4	1. Modify ***Table 25-4. NOR / PSRAM controller timing parameters*** in the EXMC chapter. 2. Modify the description of bit15 of ***Transfer status register 0 (I2C_STAT0)*** in the I2C chapter. 3. Modify ***Figure 21-27***, ***Figure 21-28*** and ***Figure 21-35*** in the I2C chapter. 4. Modify the offset address of ***Product ID register (FMC_PID)*** in the FMC chapter. 5. Modify the parameters of ***Table 16-1. Min/max FWDGT timeout period at 40 kHz (IRC40K)*** in the WDGT chapter. 6. Delete the note 2 of ***CAN0_AF remapping*** in the GOIP chapter, and modify the bit15 description of ***AFIO port configuration register 0 (AFIO_PCF0)***. 7. Modify the reset values of ***SQPI Initial Register (SQPI_INIT)*** and ***SQPI Write Command Register (SQPI_WCMD)*** in the SQPI chapter. 8. Delete the unsupported conditions of ***Table 10-6. The condition of UDRF and OVRF flag in Mode 6*** in the TMU chapter 6. 9. Add the description of bit16 of ***Additional Clock configuration register (RCU_ADDCFG)*** in the RCU chapter. 10. Added the precautions about IRC48M in the ***USBHS PHY selection, clocks and working modes*** in USBHS chapter.	Dec.16, 2021

	11. Modify the description of DIFCTL[14:0] of ***Differential mode control register (ADC_DIFCTL)*** in the ADC chapter. 12. Adjust the position of the classification description of the chip HD, XD and CL, refer to On-chip flash memory overview.	
1.5	1. Update ***Power management unit (PMU)*** chapter. 2. Update ***Serial peripheral interface/Inter-IC sound (SPI/I2S)*** chapter. 3. Update ***Inter-integrated circuit interface (I2C)*** chapter. 4. Update ***Direct memory access controller (DMA)*** chapter. 5. Update ***Comparator (CMP)*** chapter. 6. Update ***General-purpose I/Os (GPIO)*** chapter. 7. Update ***Reset and clock unit (RCU)*** chapter. 8. Update ***Universal synchronous asynchronous receiver transmitter (USART)*** chapter. 9. Update ***Analog to digital converter (ADC)*** chapter. 10. Update ***Timer (TIMERx)*** chapter. 11. Update ***Watchdog timer (WDGT)*** chapter. 12. Delete the description of USB and SOF from the ***Clock trim controller (CTC)*** chapter. 13. Update the name of ***MAC PHY data register (ENET_MAC_PHY_DATA)***. 14. Modify the description of ***NAND Flash ECC calculation module*** in EXMC chapter. 15. Delete the TRACE_MODE bit of ***Control register (DBG_CTL)*** in DBG chapter. 16. Delete the 16-bit programming of ***Characteristics*** in the FMC chapter and only describe it in ***Main flash programming***. 17. Delete the description of the extra-density (GD32E50x_XD) series from the documentation.	Jul.18, 2022
1.6	1. Update ***Clock trim controller (CTC)*** chapter. 2. Update ***Universal synchronous/asynchronous receiver /transmitter (USART)*** chapter. 3. Update ***Ethernet (ENET)*** chapter. 4. Add the ***Table 26 3. CAN Event / Interrupt flags*** in CAN chapter. 5. Update ***Interrupt/event controller (EXTI)*** chapter. 6.Modify TIMER2_CH0/TIMER2_CH1 remap pin of ***Table 8 5. TIMERx alternate function remapping*** in GPIO chapter.	Dec.22, 2022
1.7	1. Modify the description of ***VDDA domain*** in PMU chapter. 2. Modify ***IBUS*** and ***DBUS*** to ***CBUS*** and ***SBUS***。 3. Modify the description of ***Figure 19 46. Trigger to ADC selection overview*** in SHRTIMER chapter.	Jun.20, 2023

1.8	1. Modify the description of ***Table 8 10. CAN0/1 alternate function remapping*** in GPIO chapter. 2. Modify the description of ***Figure 4 1. Power supply overview*** in PMU. 3. Modify the description of ***Figure 15 1. CMP block diagram of GD32E50x series*** in CMP.	Nov.15, 2023
1.9	1. Format modification. 2. Add description in ***19.4.1 Master_TIMER unit/update event and shadow registers*** and ***19.4.2. Slave_TIMERx(x=0...4) Unit/Update event and shadow registers*** section. 3. Modify ***Figure 5 6. Clock tree*** in RCU chapter. 4. Modify the ***maximum delay generated by the analog filter is changed from 260ns to 160ns*** in I2C chapter. 5. Modify the ***Figure 19 48. DMA mode operation flowchart*** in 19.4.12.DMA mode section. 6. Modify ***Figure 27 8. Transmit descriptor in enhanced mode*** and ***Figure 27 10. Receive descriptor in enhanced mode*** in ENET chapter. 7. Delete ***BOR-related information*** in the FMC, PMU, and RCU chapters. 8. Modify the ***15.3.2. CMP I/O configuration description*** of the CMP chapter.	Jul.05, 2024
2.0	1. Modify Section ***13.4.6 Operation Modes*** to add the description: "Note: After EOC is set, a delay of one CK_ADC is required before reading the ADC conversion result." 2. Modify Section 5.2.1 in the RCU chapter to add a description of ***the clock selection for SHRTIMER in high-precision mode***. 3. In Section ***13.4.13 Programmable resolution (DRES)***, change the unit of tADC to ns. 4. Delete the description of ***the Vhyst voltage value*** in the PMU chapter. 5. Add the SHRTIMERSEL bit to ***the RCU_CFG1 register*** in Section 5.3.12. 6. Modify Sections ***19.4.1 Counter clock*** and ***19.4.2 Counter clock*** to add a description of the differences between the two clock sources, and add a fault interrupt description in Section ***19.4.10 Interrupt***. Modify the SHRTIMER_INTEN[4:0] register bit description in Section ***19.5.3 Common Registers***. 7. Modify Section ***28.6.1 USB endpoints*** and ***Table 28.3 Double buffer usage*** Description in the USB chapter, and delete the VBUS content in ***28.3. Block diagram***, ***28.4. Signal description*** and ***28.6.3. USB events and interrupts***.	Dec.25, 2024

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.