PY32F403

32-bit ARM® Cortex® M4F Microcontroller

Reference manual

Puya Semiconductor (Shanghai) Co., Ltd

Contents

1. Introduction

The system consists of the Cortex-M4 processor core, which includes SIMD's DSP enhancements and the Floating Process Unit (FPU) unit, which can be switched on or off depending on the configuration, but is supported by default for SIMD instructions. The on-chip Flash is used only as a read-only space for software instructions and data during system operation, with an instruction and data cache added to the system to increase the efficiency of the CM4, and an on-chip Flash with 128-bit data to reduce access time to the on-chip Flash. To reduce access time to the on-chip Flash, the on-chip Flash provides a 128-bit data read bit width. The Quad/Octal SPI Flash extends the system's non-volatile runnable storage while occupying only a small number of GPIO ports. The system includes DMA1 and 2 which can support up to 12 channels for direct data transfer between memory, peripherals including ADC , SDIO, I2S, I2C and USART.

The system includes two peripheral system buses, APB1 and APB2, with the same speed as the AHB bus. The peripherals include 17 timer units and 13 communication interface units. The former includes ten 16-bit timers, two 16-bit timers with PWM control, two watchdog timers, two basic timers and one systick timer. The latter includes two I2C interfaces, five USART interfaces, three SPI interfaces, a CAN bus controller, a USB 2.0 interface supporting full speed operation and an SDIO interface.

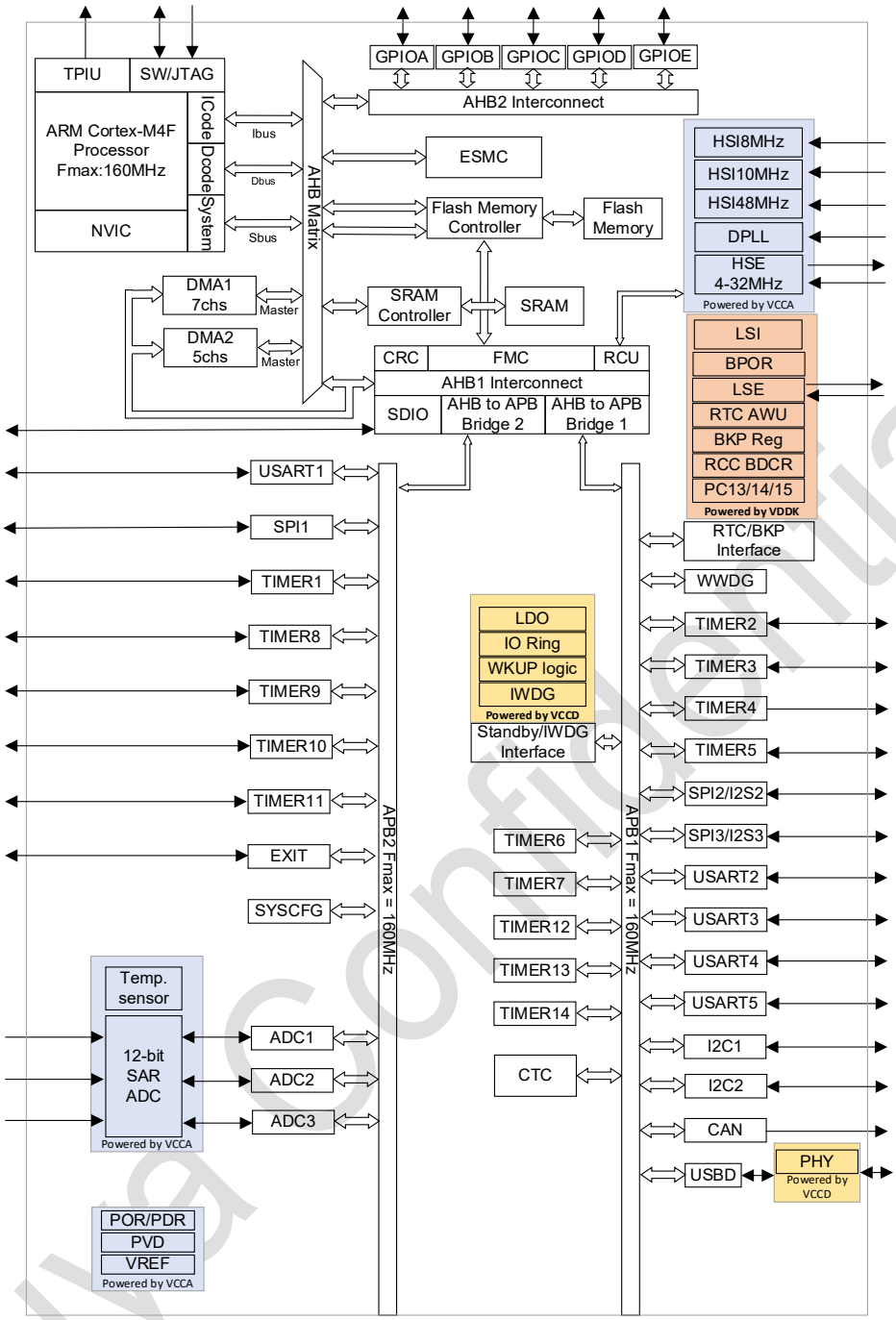


Figure 1-1 Module block diagram

2. System and memory architecture

The PY32F403XX family of devices is a 32-bit general purpose microcontroller based on the Arm® Cortex®-M4 processor. The Arm® Cortex®-M4 processor includes three AHB buses called the I-CODE bus, the D-Code bus and the system bus respectively. All memory accesses to the Cortex®-M4 processor are performed on these three buses, depending on the purpose and target memory space. The organisation of the memory is based on a Harvard structure, with pre-defined memory mappings and up to 4 GB of storage space, which fully guarantees system flexibility and scalability.

2.1. Arm® Cortex®-M4 Processor Introduction

The Cortex®-M4 processor is a 32-bit processor with low interrupt latency times and low cost debug features. The high level of integration and enhanced features make the Cortex™-M4 processor suitable for those market segments that require high performance and low power microcontrollers. The Cortex®-M4 processor is based on the Armv7 architecture and supports a powerful and scalable instruction set including general purpose data processing I/O control tasks, enhanced data processing bit field operations, DSP (Digital Signal Processing) and floating point instructions. Some of the system peripherals provided by the Cortex®-M4 are listed below:

- Internal bus matrix for interconnecting the I-Code bus, D-Code bus, system bus, private bus (PPB) and debug-only bus (AHB-AP);
- Nested Vector Interrupt Controller (NVIC);
- Flash Address Reload and Breakpoint Unit (FPB);
- Data Watchpoint and Trace Unit (DWT);
- Instruction Trace Macro Unit (ITM);
- Embedded Trace Macro Unit (ETM);
- Serial Line and JTAG Debug Interface (SWJ-DP);
- Trace Port Interface Unit (TPIU);
- Memory Protection Unit (MPU);
- Floating Point Unit (FPU).

2.2. System architecture

The system uses a 32-bit multi-layer AHB bus matrix that allows for parallel communication between multiple masters and multiple slaves:

- Five master buses:
  - Cortex™-M4F core I-bus, D-bus and S-bus
  - DMA1 memory bus
  - DMA2 memory bus
- Five Controlled Buses:
  - Internal Flash ICode bus
  - Internal Flash DCode bus
  - Internal SRAM bus
  - AHB1 peripheral bus (including AHB to APB bus bridge and APB peripherals)
  - AHB2 peripheral bus

- ESMC

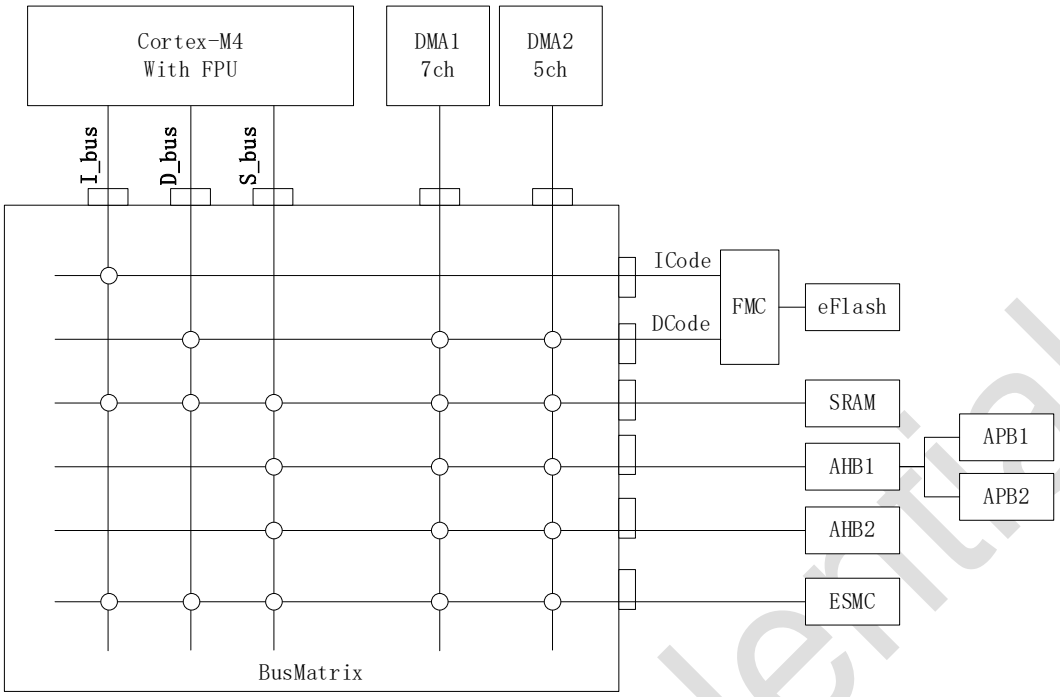


Figure 2-1 Bus Architecture

2.2.1. I_bus

This bus is used to connect the instruction bus of the Cortex™-M4F core to the bus matrix. The kernel gets its instructions via this bus. This bus accesses the memory containing the code (internal Flash/SRAM or external memory via ESMC).

2.2.2. D_bus

This bus is used to connect the Cortex™-M4F data bus to the bus matrix. The kernel performs immediate number loading and debug accesses via this bus. This bus accesses memory containing code or data (internal Flash/SRAM or external memory via ESMC).

2.2.3. S_bus

This bus is used to connect the system bus of the Cortex™-M4F core to the bus matrix. This bus is used to access data located in peripherals or SRAM. This bus accesses the 64 KB internal SRAM, the APB1 peripheral including the AHB peripheral, the APB2 peripheral and the external memory via the ESMC.

2.2.4. DMA1/DMA2 BUS

This bus is used to connect the main DMA memory bus interface to the bus matrix. The DMA accesses via this bus are the 64KB internal SRAM, the APB1 peripheral including the AHB peripheral, the APB2 peripheral and the external memory via the ESMC.

2.2.5. Bus Matrix

The bus matrix is used to manage the arbitration of accesses between master buses. Arbitration is performed using a round-robin scheduling algorithm. The system consists of five Master devices, I_BUS, D_BUS, S_BUS, DMA1 and DMA2 of the ARM® Cortex™-M4, and four Slave devices, internal

Flash memory, internal SRAM, AHB peripherals (AHB2APB Bridge 1 for the peripherals and AHB2APB Bridge 2 for the AHB2APB Bridge 1 and AHB2APB Bridge 2), and ESMC.

2.2.6. AHB/APB bus bridge

With the AHB/APB bus bridge, a fully synchronous connection between the AHB bus and the APB bus is possible, allowing flexible selection of peripheral frequencies.

After each chip reset, all peripheral clocks are switched off (except for the SRAM and Flash interfaces). Before a peripheral can be used, its clock must be enabled in the RCC_AHBxENR or RCC_APBxENR registers.

2.3. Memory Organisation

2.3.1. Introduction

The Arm® Cortex®-M4 processor has a Harvard architecture that allows the use of mutually independent buses for reading instructions and loading/storing data. Instruction code and data are located in the same memory address space, but in different address ranges. Program memory, data memory, registers and I/O ports are all within the same linear 4 GB address space.

The individual bytes are encoded in memory in a small-end format. The lowest numbered byte in a word is considered the lowest valid byte for that word, while the highest numbered byte is considered the highest valid byte.

For more information on the mapping of peripheral registers, see the relevant section.

The addressable memory space is divided into 8 main blocks of 512 MB each.

2.3.2. Memory Map

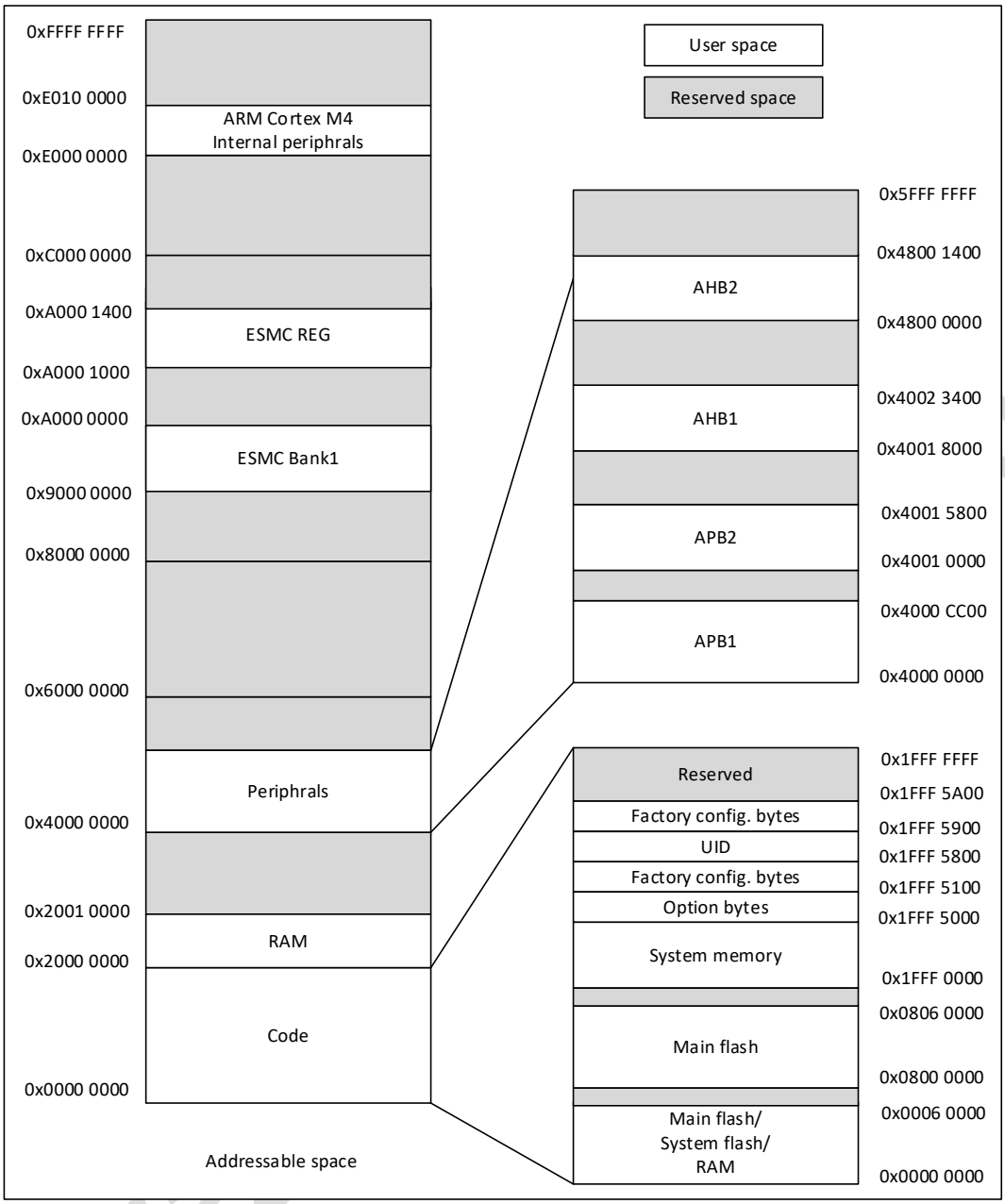


Figure 2-2 Memory Map

All other memory space not allocated to on-chip memory and peripherals is reserved address space. Refer to the table below for detailed memory and register space mapping.

The table below gives the peripherals available in the product and the corresponding address boundaries.

Table 2-1 Peripheral register address

Memory start/stop address	Peripheral	Bus	Register map
0xA000 1000 - 0xA000 13FF	ESMC	AHB	
0x4800 1400 - 0x5FFF FFFF	Reserve	AHB2	
0x4800 1000 - 0x4800 13FF	GPIOE		
0x4800 0C00 - 0x4800 0FFF	GPIOD		
0x4800 0800 - 0x4800 0BFF	GPIOC		
0x4800 0400 - 0x4800 07FF	GPIOB		

Memory start/stop address	Peripheral	Bus	Register map
0x4800 0000 - 0x4800 03FF	GPIOA		
0x4002 3400 - 0x47FF FFFF	Reserve	AHB1	
0x4002 3000 - 0x4002 33FF	CRC		
0x4002 2400 - 0x4002 2FFF	Reserve		
0x4002 2000 - 0x4002 23FF	FMC		
0x4002 1400 - 0x4002 1FFF	Reserve		
0x4002 1000 - 0x4002 13FF	RCC		
0x4002 0800 - 0x4002 0FFF	Reserve		
0x4002 0400 - 0x4002 07FF	DMA2		
0x4002 0000 - 0x4002 03FF	DMA1		
0x4001 8400 - 0x4001 FFFF	Reserve		
0x4001 8000 - 0x4001 83FF	SDIO		
0x4001 5800 - 0x4001 7FFF	Reserve	APB2	
0x4001 5400 - 0x4001 57FF	TIMER11		
0x4001 5000 - 0x4001 53FF	TIMER10		
0x4001 4C00 - 0x4001 4FFF	TIMER9		
0x4001 4000 - 0x4001 4BFF	Reserve		
0x4001 3C00 - 0x4001 3FFF	ADC3		
0x4001 3800 - 0x4001 3BFF	USART1		
0x4001 3400 - 0x4001 37FF	TIMER8		
0x4001 3000 - 0x4001 33FF	SPI1		
0x4001 2C00 - 0x4001 2FFF	TIMER1		
0x4001 2800 - 0x4001 2BFF	ADC2		
0x4001 2400 - 0x4001 27FF	ADC1		
0x4001 0800 - 0x4001 23FF	Reserve		
0x4001 0400 - 0x4001 07FF	EXTI		
0x4001 0000 - 0x4001 03FF	SYSCFG		
0x4000 CC00 - 0x4000 FFFF	Reserve	APB1	
0x4000 C800 - 0x4000 CBFF	CTC		
0x4000 7800 - 0x4000 C7FF	Reserve		
0x4000 7400 - 0x4000 77FF	Reserve		
0x4000 7000 - 0x4000 73FF	PWR		
0x4000 6C00 - 0x4000 6FFF	BKP		
0x4000 6800 - 0x4000 6BFF	Reserve		
0x4000 6400 - 0x4000 67FF	CAN		
0x4000 6000 - 0x4000 63FF	USBD/CAN share 512 bytes of SRAM		
0x4000 5C00 - 0x4000 5FFF	USBD		
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 53FF	UASRT5		
0x4000 4C00 - 0x4000 4FFF	UASRT4		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		

Memory start/stop address	Peripheral	Bus	Register map
0x4000 4000 - 0x4000 43FF	Reserve		
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S		
0x4000 3800 - 0x4000 3BFF	SPI2/I2S		
0x4000 3400 - 0x4000 37FF	Reserve		
0x4000 3000 - 0x4000 33FF	IWDG		
0x4000 2C00 - 0x4000 2FFF	WWDG		
0x4000 2800 - 0x4000 2BFF	RTC		
0x4000 2400 - 0x4000 27FF	Reserve		
0x4000 2000 - 0x4000 23FF	TIMER14		
0x4000 1C00 - 0x4000 1FFF	TIMER13		
0x4000 1800 - 0x4000 1BFF	TIMER12		
0x4000 1400 - 0x4000 17FF	TIMER7		
0x4000 1000 - 0x4000 13FF	TIMER6		
0x4000 0C00 - 0x4000 0FFF	TIMER5		
0x4000 0800 - 0x4000 0BFF	TIMER4		
0x4000 0400 - 0x4000 07FF	TIMER3		
0x4000 0000 - 0x4000 03FF	TIMER2		

2.4. Embedded SRAM

The PY32F403XX series devices have a built-in static SRAM of up to 64K bytes, the SRAM capacity may vary depending on the product definition, please refer to the Datasheet for details. it can be accessed as byte, half word (16 bits) or full word (32 bits). the starting address of the SRAM is 0x2000 0000.

2.5. Embedded FLASH

The Flash memory consists of two distinct physical areas:

- Main flash area, 384KBytes, which contains application programs and user data. The Flash capacity may vary depending on the product definition, please refer to the Datasheet for details. Software access to the space outside the set range will result in a hard fault.
- Information area, 24Kbytes, which contains the following parts:
  - Factory config. bytes: 3328Bytes, used to store trimming data (including HSI trimming data), flash size configuration information, and power-up read checksum.
  - Factory config. bytes:
    chip power-up read parity code
    Chip hardware trimming configuration value
    Device ID code
  - UID: 512 bytes, used to store the chip's UID
  - Option bytes: 256 bytes for chip hardware and memory protection configuration values
  - System memory: 20KBytes for the boot loader

The Flash interface implements instruction reading and data access based on the AHB protocol, it also implements the basic program/erase operations of the flash through registers.

2.6. Bit Segments

The Cortex™-M4F memory map consists of two-bit segment regions. These regions map each word in the memory alias region to the corresponding bit in the memory bit segment region. Writing a word in the alias region is equivalent to performing a read-modify-write operation on the target bit in the bit segment region.

Peripheral registers and SRAM are mapped to a bit-segment area, allowing read and write operations to be performed on a single bit-segment. These operations are only available for Cortex™-M4F accesses and are not available for other bus master interfaces such as DMA.

The correspondence between each word in the alias area and each bit in the bit segment area can be illustrated by a mapping formula. The mapping formula is as follows.

bit_word_addr = bit_band_base + (byte_offset x 32) + (bit_number × 4)

- bit_word_addr represents the address of the word that will be mapped to the target bit in the alias area
- bit_band_base represents the start address of the alias area
- byte_offset represents the byte number in the band area where the target bit is located
- bit_number represents the bit position of the target bit (0-7)

Example

The following example shows how bit 2 of the byte at SRAM address 0x20000300 can be mapped to the alias area:

0x22006008 = 0x22000000 + (0x300*32) + (2*4)

A write operation to address 0x22006008 is equivalent to a read-modify-write operation on bit 2 of the byte at SRAM address 0x20000300.

A read operation on address 0x22006008 returns the value of bit 2 of the byte at SRAM address 0x20000300 (0x01 for a position bit, 0x00 for a bit reset).

See the Cortex™-M4F Programming Manual for more information on bit segments。

2.7. Start-up configuration

In the PY32F403xx, three different boot modes can be selected via the BOOT[1:0] pins, as shown in the table below.

Table 2-2 BOOT pin configuration

Boot Mode	Description	Pin Configuration	
		BOOT1	BOOT0
Main flash memory	The main flash memory is selected as the boot area	X	0
System memory	System memory selected as boot area	0	1
On-chip SRAM	Internal SRAM selected as boot area	1	1

After a system reset, the value of the BOOT pin will be latched. The user can select the boot mode after a reset by setting the status of the BOOT1 and BOOT0 pins. When exiting from standby mode, the value of the BOOT pin will be relocked. Therefore, in standby mode the BOOT pin should be left in the desired boot configuration.

Depending on the selected boot mode, the main flash memory, system memory or SRAM can be accessed as follows

- Boot from main flash memory: the main flash memory is mapped to the boot space (0x0000 0000) but it can still be accessed at its original address (0x0800 0000), i.e. the contents of the flash memory can be accessed in two address areas, 0x0000 0000 or 0x0800 0000.
- Boot from system memory: the system memory is mapped to the boot space (0x0000 0000) but it can still be accessed at its original address (0x1FFF B000).
- Boot from internal SRAM: The SRAM is mapped to the boot space (0x0000 0000) but can still be accessed at its original address (0x2000 0000).

After the boot delay, the CPU gets the address at the top of the stack from address 0x0000 0000 and starts executing the code from the address indicated by 0x0000_0004 in the boot memory. Because of the fixed memory image, the code area always starts at address 0x0000_0000 (accessed via the ICode and DCode buses), while the data area (SRAM) always starts at address 0x2000_0000 (accessed via the system bus). the Cortex M4F CPU always gets the reset vector from the ICode bus, i.e. the boot is only suitable for starting from the The PY32F403xx family of microcontrollers implements a special mechanism for booting from the on-chip SRAM. When booting from the on-chip SRAM, the vector table must be remapped in the SRAM using the NVIC's exception table and offset registers in the initialisation code of the application.

2.7.1. Embedded start-up program

The system memory contains an embedded bootloader which can be used to program the flash memory. The bootloader is used to reprogram the flash memory via USART1, USART2 or the USB interface.

2.7.2. Physical remapping

When the boot pin is selected, the application software can set certain memories to be accessed from the code space (so that code can be executed via the ICode bus instead of the system bus). Such a modification is achieved by programming SYSCFG_MEMRMP in the SYSCFG controller. The following memories can thus be remapped:

- Main Flash
- System memory
- Embedded SRAM
- ESMC

Table 2-3 Port Bit Configuration Table

Address	Boot/remap in main Flash	Boot/remap in embedded SRAM	Boot/remap in system memory	Remapping in ESMC
0x2000 0000 - 0x2000 FFFF	SRAM	SRAM	SRAM	SRAM
0x1FFF 0000 - 0x1FFF 4FFF	System memory	System memory	System memory	System memory
0x0808 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved	Reserved
0x0800 0000 - 0x0807 FFFF	Flash	Flash	Flash	Flash
0x0006 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved	ESMC
0x0000 0000 - 0x0005 FFFF	Flash Use alias	SRAM Use alias	System memory (20KB) Use alias	ESMC

3. Embedded Flash Memory

3.1. Introduction

The Flash interface manages CPU access to the Flash via the AHB I-Code and D-Code. The interface performs erase and program operations on the Flash and implements read and write protection mechanisms.

The Flash interface accelerates code execution by means of instruction prefetching and caching mechanisms.

3.1.1. Main features

- Memory organisation
  - Main memory: 384 Kbytes max.
  - Information memory: 24 Kbytes
  - Page size: 256 bytes
  - Sector size: 2 Kbytes
  - Block size: 32 Kbytes
  - 128-bit wide data read
  - 32-bit wide data write
- Flash read operation
- Flash programming operation (page programming)
- Flash erase operation (page erase/sector erase/block erase/full erase)
- Read/write protection
- Prefetch operations on I-Code
- Branch Cache on I-Code
- Data cache on D-Code
- Option byte load

3.1.2. Module block diagram

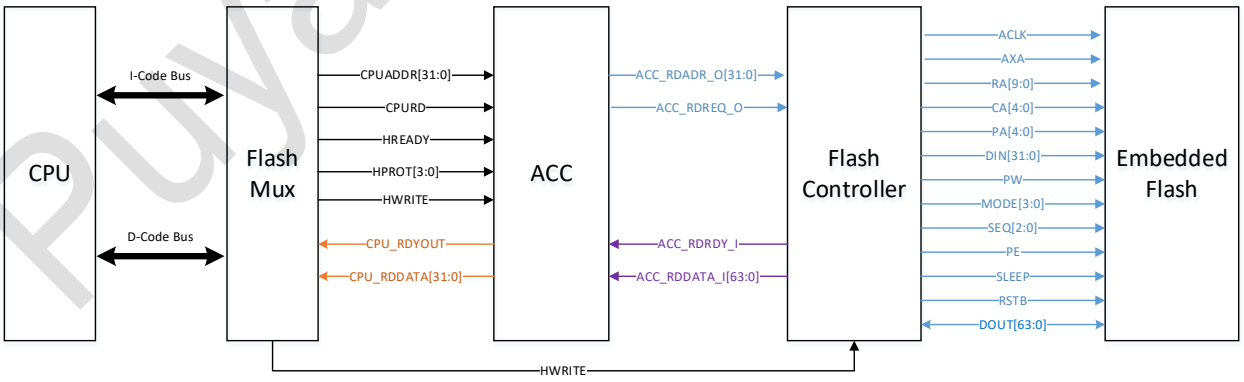


Figure 3-1 Module block diagram

3.2. Functional description

3.2.1. Flash memory structure

Flash memory consists of 128-bit wide memory cells that can be used for program and data storage, with a Page size of 256Bytes, a Sector size of 2Kbytes and a Block size of 32KBytes.

Functionally, Flash memory is divided into Main Flash, which has a maximum capacity of 384Kbytes, and information flash, which has a capacity of 24Kbyte, as follows:

Table 3-1 Flash memory structure and boundary address

Area	Space Name			Address	Size (bytes)
Main memory	Block 0	Sector 0	Page 0-7	0x0800 0000 - 0x0800 07FF	2 K
		Sector 1	Page 8-15	0x0800 0800 - 0x0800 0FFF	2 K
		Sector 2	Page16-23	0x0800 1000 - 0x0800 17FF	2 K
		Sector 3	Page 24-31	0x0800 1800 - 0x0800 1FFF	2 K
		Sector 4	Page 32-39	0x0800 2000 - 0x0800 27FF	2 K
		Sector 5	Page 40-47	0x0800 2800 - 0x0800 2FFF	2 K
		Sector 6	Page 48-55	0x0800 3000 - 0x0800 37FF	2 K
		Sector 7	Page 56-63	0x0800 3800 - 0x0800 3FFF	2 K
		Sector 8	Page 64-71	0x0800 4000 - 0x0800 47FF	2 K
		Sector 9	Page 72-79	0x0800 4800 - 0x0800 4FFF	2 K
		Sector 10	Page 80-87	0x0800 5000 - 0x0800 57FF	2 K
		Sector 11	Page 88-95	0x0800 5800 - 0x0800 5FFF	2 K
		Sector 12	Page 96-103	0x0800 6000 - 0x0800 67FF	2 K
		Sector 13	Page 104-111	0x0800 6800 - 0x0800 6FFF	2 K
		Sector 14	Page 112-119	0x0800 7000 - 0x0800 77FF	2 K
		Sector 15	Page 120-127	0x0800 7800 - 0x0800 7FFF	2 K
	…	…	…	…	…
	Block 11	Sector 176	Page 1408-1415	0x0805 8000 - 0x0805 87FF	2 K
		Sector 177	Page 1416-1423	0x0805 8800 - 0x0805 8FFF	2 K
		Sector 178	Page 1424-1431	0x0805 9000 - 0x0805 97FF	2 K
		Sector 179	Page 1432-1439	0x0805 9800 - 0x0805 9FFF	2 K
		Sector 180	Page 1440-1447	0x0805 A000 - 0x0805 A7FF	2 K
		Sector 181	Page 1448-1455	0x0805 A800 - 0x0805 AFFF	2 K
		Sector 182	Page 1456-1463	0x0805 B000 - 0x0805 B7FF	2 K
		Sector 183	Page 1464-1471	0x0805 B800 - 0x0805 BFFF	2 K
		Sector 184	Page 1472-1479	0x0805 C000 - 0x0805 C7FF	2 K
		Sector 185	Page 1480-1487	0x0805 C800 - 0x0805 CFFF	2 K
		Sector 186	Page 1488-1495	0x0805 D000 - 0x0805 D7FF	2 K
		Sector 187	Page 1496-1503	0x0805 D800 - 0x0805 DFFF	2 K
		Sector 188	Page 1504-1511	0x0805 E000 - 0x0805 E7FF	2 K
		Sector 189	Page 1512-1519	0x0805 E800 - 0x0805 EFFF	2 K
		Sector 190	Page 1520-1527	0x0805 F000 - 0x0805 F7FF	2 K
		Sector 191	Page 1528-1535	0x0805 F800 - 0x0805 FFFF	2 K
Informatin	System memory			0x1FFF 0000 - 0x1FFF 4FFF	20 K

Area	Space Name	Address	Size (bytes)
block	Option bytes	0x1FFF 5000 - 0x1FFF 50FF	256
	Factory config. bytes	0x1FFF 5100 - 0x1FFF 51FF	256
	Factory config. bytes	0x1FFF 5200 - 0x1FFF 52FF	256
	Factory config. bytes	0x1FFF 5300 - 0x1FFF 53FF	256
	Factory config. bytes	0x1FFF 5400 - 0x1FFF 54FF	256
	Factory config. bytes	0x1FFF 5500 - 0x1FFF 55FF	256
	HSI8M Trim	0x1FFF 5600 - 0x1FFF 56FF	256
	Factory config. bytes	0x1FFF 5700 - 0x1FFF 57FF	256
	UID	0x1FFF 5800 - 0x1FFF 58FF	256
	Factory config. bytes	0x1FFF 5900 - 0x1FFF 59FF	256
	Reserve	0x1FFF 5A00 - 0x1FFF 5FFF	1.5 K

3.2.2. Flash memory read operations and access latencies

In order to read instructions/data from Flash memory correctly, the number of wait cycles (Latency) of the register FLASH_ACR needs to be set according to the frequency of the CPU clock (HCLK).

Table 3-2 Number of wait states according to CPU clock (HCLK) frequency

HCLK(MHz)	Wait states(Latency)
Fhclk <= 28MHz	0 (1 HCLK cycles)
28MHz < Fhclk <= 60MHz	1 (2 HCLK cycles)
60MHz < Fhclk <= 90MHz	3 (4 HCLK cycles)
90MHz < Fhclk <= 120MHz	4 (5 HCLK cycles)
120MHz < Fhclk <= 140MHz	5 (6 HCLK cycles)
140MHz < Fhclk	6 (7 HCLK cycles)

Note: In the case of latency 0, if the HPRE divider is used, the clock cannot be higher than 60MHz before the divider.

After a chip reset (including power-on reset, system reset and exit standby reset), the HCLK clock frequency is 8MHz and the Flash read cycle count is 0.

When changing the CPU clock frequency or range, the following software steps must be followed to adjust the number of wait cycles required to access the Flash memory.

- Steps to follow when increasing CPU frequency

1. program the new wait cycle number into the LATENCY bit in the FLASH_ACR register
2. Check if the new wait cycle is set successfully by reading the FLASH_ACR register
3. modify the CPU clock source by rewriting the SW bit in the RCC_CFGR register
4. If necessary, modify the CPU clock prescaler by overwriting the HPRE bit in RCC_CFGR
5. Check whether the new CPU clock source and/or the new CPU clock prescaler is successfully set by reading the corresponding clock source status (SWS bit) and/or AHB prescaler value (HPRE bit) in the RCC_CFGR register

- Steps to follow when lowering the CPU frequency

1. Modify the CPU clock source by overwriting the SW bit in the RCC_CFGR register
2. If necessary, modify the CPU clock prescaler by overwriting the HPRE bit in the RCC_CFGR register
3. check whether the new CPU clock source and/or the new CPU clock prescaler is successfully set by reading the corresponding clock source status (SWS bit) and/or AHB prescaler value (HPRE bit) in the RCC_CFGR register
4. program the new number of wait cycles into the LATENCY bit in FLASH_ACR
5. check if the new wait cycle is set successfully by reading the FLASH_ACR register

3.2.3. Adaptive Real-Time Memory Accelerator™ (ART Accelerator™)

The proprietary Adaptive Real-Time (ART) Memory Accelerator is optimised for the ARM® Cortex™-M4 processor. The accelerator takes advantage of the inherent performance benefits of the ARM Cortex M4F, overcoming the usual conditions where high-speed processors often have to wait for FLASH reads during operation.

To exploit the full performance of the processor, the accelerator will implement instruction pre-fetch queues and branch caching, thereby increasing the speed of program execution on 128-bit Flash. According to CoreMark benchmark tests, the performance achieved with the ART accelerator is equivalent to Flash executing a program with 0 wait cycles at CPU frequencies up to 160 MHz.

- **instruction prefetch**

Each Flash read operation can read 128 bits, either 4 lines of 32-bit instructions or 8 lines of 16-bit instructions, depending on the program burned into the Flash. Therefore, for sequentially executed code, at least 4 CPU cycles are required to execute the 128-bit instruction line from the previous read. When the CPU requests the current instruction line, the next successive 128-bit instruction line in Flash can be read using the prefetch operation of the I-Code bus. The prefetch function can be enabled by setting the PRFTEN bit in the FLASH_ACR register to 1. This is useful when at least one wait cycle is required to access the Flash.

- **instruction cache memory**

To reduce the time loss due to instruction hops, 64 lines of 128-bit instructions can be stored in the instruction cache memory. This feature can be enabled by setting the Instruction Cache Enable (ICEN) in the FLASH_ACR register to position 1. Whenever an instruction miss occurs (i.e. the requested instruction does not exist in the currently used instruction line, prefetched instruction line or instruction cache memory), the system will copy the newly read line into the instruction cache memory. If the instruction requested by the CPU already exists in the instruction cache, it can be fetched immediately without any delay. When the instruction cache is full, the LRU (least recently used) policy can be used to determine the instruction lines to be replaced in the instruction cache. This feature is ideal for code that contains loops.

- **Data Management**

During the CPU flow execution phase, the data buffer pool in Flash is accessed via the D-Code bus. Therefore, the CPU flow does not continue until the requested data has been provided. In order to reduce the resulting time loss, accesses via the AHB data bus D-Code take precedence over accesses via the AHB instruction bus I-Code. If certain data is used frequently, the data cache memory can be enabled by setting the Data Cache Enable (DCEN) in the FLASH_ACR register to position 1. This

feature works similarly to the instruction cache memory, but the size of the data retained is limited to 8 lines of 128 bits/line.

3.2.4. Erasing and programming operations

The flash can be programmed via ICP or IAP.

ICP: is used to update the contents of the entire Flash memory, either using the SWD protocol or boot loader, to load the user application into the MCU. ICP provides fast and efficient design iterations and eliminates unnecessary packet handling or socketing.

IAP: The data to be programmed can be downloaded to flash using the communication interface supported by the chip. IAP allows the user to reprogram the flash memory while the application is running. The flash memory will then already contain part of the application previously programmed in using ICP.

If a reset occurs during a flash programming or erase operation, the contents of the flash memory are not protected. Any read of the flash during a program or erase operation will delay the bus. As soon as the programming or erase operation is completed, the read operation can be performed correctly. This also means that code and data cannot be read while a program or erase operation is in progress.

For programming and erasing operations, the HSI must be switched on and the user is advised to perform programming and erasing operations at the default clock to prevent problems.

- **Flash Unlock**

After a reset, the Flash control register (FLASH_CR) is not allowed to perform write operations to prevent accidental operations on the Flash due to electrical interference etc. Each erase and program operation to the Flash must Unlock the timing by writing FLASH_KEYR to enable access to the FLASH_CR register.

The specific steps are as follows:

Step 1: Write KEY1=0x4567 0123 to the FLASH_KEYR register

Step 2: Write KEY2=0xCDEF 89AB to the FLASH_KEYR register Any incorrect timing will lock the FLASH_CR register until the next reset. At incorrect KEY timing, a bus error is detected and a Hard Fault interrupt is generated. Such errors include a KEY1 mismatch on the first write cycle, or a KEY1 match but a KEY2 mismatch on the second write cycle.

The FLASH_CR register can be re-locked by a software write to the LOCK bit of the FLASH_CR register.

Note: When the BSY bit of the FLASH_SR register is set, the FLASH_CR register cannot be written to. At this point, any attempt to write to this register (FLASH_CR) will cause a delay on the AHB bus until the BSY bit is cleared.

3.2.5. Flash erase operations

Flash erase operations support page,sector,mass erase, mass erase does not work on the information memory.

- **Flash Page Erase**

Page erase is used to erase 256bytes of main flash, but does not work on the information area.

When a page is protected by WRP, it will not be erased and the WRPERR bit will be set.

The page erase operation is performed as follows:

1) Check the BSY bit of the FLASH_SR register to make sure there are no flash operations in progress

2) Write KEY1 and KEY2 to the FLASH_KEYR register in sequence to unprotect the FLASH_CR register

3) Set the PER and EOPIE bits of the FLASH_CR register

4) Write any data to the page (must be 32 bits data)

5) Wait for the BSY bit to be cleared

6) Check that the EOP flag bit is set

7) Clear the EOP flag

- Flash sector erase

Sector erase is used to erase 2Kbytes of main flash, but does not work on the information sector. When a sector is protected by WRP, it is not erased and the WRPERR bit is set.

The sector erase operation is performed as follows:

1) Check the BSY bit to make sure there are no Flash operations in progress

2) Write KEY1 and KEY2 to the FLASH_KEYR register in sequence to unprotect the FLASH_CR register

3) Set the SER and EOPIE bits of the FLASH_CR register

4) Write any data to this sector

5) Wait for the BSY bit to be cleared

6) Check that the EOP flag bit is set

7) Clear the EOP flag

- **Flash Block Erase**

Block erase is used to erase 32Kbytes of main flash, but does not work on the information sector. When a block is protected by WRP, it is not erased and the WRPERR bit is set.

The steps for a block erase operation are as follows:

1) Check the BSY bit to make sure there are no Flash operations in progress

2) Write KEY1 and KEY2 to the FLASH_KEYR register in sequence to unprotect the FLASH_CR register

3) Set the BER and EOPIE bits of the FLASH_CR register

4) Write any data to the block

5) Wait for the BSY bit to be cleared

6) Check that the EOP flag bit is set

7) Clear the EOP flag

- **flash erase**

Mass erase is used to erase the entire main flash, but not the information area. When WRP is enabled, the mass erase function is disabled, no mass erase operation is generated and the WRPERR bit is set.

The procedure for a mass erase operation is as follows:

1) Check the BSY bit to verify that there are no Flash operations in progress

2) Write KEY1,KEY2 to the FLASH_KEYR register in sequence to unprotect the FLASH_CR register

3) Set the MER and EOPIE bits of the FLASH_CR register

4) Write any data (32bit data) to any main flash space of the flash

5) Wait for the BSY bit to be cleared

6) Check that the EOP flag bit is set

7) Clear the EOP flag

3.2.6. Flash memory write operations

The Flash memory is programmed in 32-bit (word) increments (a half word or byte operation will generate a hardfault) for the entire page at a time. The programming operation starts when the PG bit of the FLASH_CR register is set and the CPU writes 32-bit data to the FLASH memory address space. If the flash address space to be programmed is an area protected by the FLASH_WRPR register, the program operation is ignored and the WRPERR bit of the FLASH_SR register is set. At the end of the programming operation, the EOP bit of the FLASH_SR register is set.

The specific flash programming procedure is shown below:

1) Check the BSY bit of the FLASH_SR register to determine if there is no flash operation currently in progress

2) If there is no flash erase or programming operation in progress, the software reads the 64 words of the page (this step is performed if the data already on the page needs to be retained, otherwise it is skipped)

3) Write KEY1 and KEY2 to the FLASH_KEYR register in sequence to unprotect the FLASH_CR register

4) Set the PG and EOPIE bits of the FLASH_CR register

5) Program word 1 to 63 to the target address (only 32-bit programs are accepted)

6) Set the PGSTRT of the FLASH_CR register

7) Write the 64th word

8) Wait for the BSY bit in the FLASH_SR register to be cleared

9) Check the EOP flag of the FLASH_SR register (when the program operation has succeeded, this bit is set) and then software clears this bit

10) If there is no more programming operation, the software clears the PG bit When step 7) above is successfully executed, the programming operation is automatically started and the BSY bit is set by hardware.

3.2.7. Flash option bytes

3.2.7.1. Description of the Flash option byte

The part of the information area of the flash inside the chip is used as an Option byte to store the hardware configuration required by the chip or the user for the application. For example, the watchdog can be selected in hardware or software mode.

For data security purposes, the option byte is stored as a body and inverse code.

Table 3-3 Option byte format

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

The contents of the Option byte can be read from the memory address described in the table Option byte organization, or from the following registers associated with the Option byte:

- FLASH user option register (FLASH_OPTR)
- FLASH WRP address register (FLASH_WRPR)

Table 3-4 Option byte organization

Word Address	Description
0x1FFF 5000	Option byte for Flash User option and its complemented
0x1FFF 5004	Reserved
0x1FFF 5008	Reserved
0x1FFF 500C	Option byte for Flash WRP address and its complemented
0x1FFF 5010	Reserved
0x1FFF 5014	Reserved
…	Reserved
0x1FFF 50FF	Reserved

Option byte for Flash User option

Flash memory address: 0x1FFF 5000

Production value:0x8F55 70AA

After the power-on reset (POR/OBL_LAUNCH), the corresponding value is read from the option bytes area of the flash information memory and written to the corresponding option bit of this register.

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Res	~nRST_STDBY	~nRST_STOP	~IWDG_SW	Res				~RDP[7:0]							
R	R	R	R	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**
Res	nRST_STDBY	nRST_STOP	IWDG_SW	Res				RDP[7:0]							
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31	Reserved	RES	
30	~nRST_STDBY	R	Inverse code of NRST_STDBY
29	~nRST_STOP	R	Inverse code of NRST_STOP
28	~IWDG_SW	R	Inverse code of IWDG_SW
27：25	Reserved	RES	
24	Reserved	RES	
23：16	~RDP	R	Inverse code of RDP
15	Reserved	RES	
14	nRST_STDBY	R	0： Reset generated when entering Standby mode 1： No reset generated
13	nRST_STOP	R	0： Reset generated when entering Stop mode 1： No reset generated
12	IWDG_SW	R	0: Hardware watchdog

			1: Software watchdog
11：9	Reserved	RES	
8	Reserved	RES	
7：0	RDP	R	0xAA：level 0, read protection inactive Non 0xAA：level 1, read protection active

Option byte for Flash WRP address

Flash memory address: 0x1FFF 500C

Production value: 0xF000 0FFF

After the power-on reset (POR/OBL_LAUNCH), the corresponding value is read from the option bytes area of the flash information memory and written to the corresponding option bit of this register.

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Res	Res	Res	Res	~WRP[11:0]											
				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**
Res	Res	Res	Res	WRP[11:0]											
				R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
27：16	Complemented WRP	R	Inverse code for WRP
11：0	WRP	R	0: block[y] is protected 1: block[y] is unprotected y=0~11

3.2.7.2. Flash Option byte write

After reset, the bits in the FLASH_CR register associated with the Option byte are write-protected. The OPTLOCK bit in the FLASH_CR register must be cleared before the operation associated with the option byte can be performed.

The following steps are used to unlock this register:

1) Unlock the FLASH_CR register from write protection by Unlock timing

2) Write OPTKEY1=0x0819 2A3B to the FLASH_OPTKEYR register

3) Write OPTKEY2=0x4C5D 6E7F to FLASH_OPTKEYR register

Any incorrect timing will lock the FLASH_CR register until the next reset. At incorrect KEY timing, a bus error is detected and a Hard Fault interrupt is generated.

The User option (option bytes of the information flash) can be protected against unwanted erase/program operations by software writing the OPTLOCK bit of the FLASH_CR register.

If the Lock bit is set by software, the OPTLOCK bit is also set automatically.

Modifying user option bytes

The program operation of option bytes is different from the operation of Main flash. In order to modify the option bytes, the following steps are required:

1) Clear the OPTLOCK bit using the procedure described previously
2) Check the BSY bit to make sure there are no ongoing flash operations
3) Write the desired value (1 to 2 words) to the option bytes register FLASH_OPTR/ FLASH_WRPR
4) Set the OPTSTRT bit and EOPIE bit of register FLASH_CR
5) Write any 32-bit data to any address in main flash (triggers a formal write operation)
6) wait for the BSY bit to be cleared
7) wait for EOP to be pulled high and software to clear

Any change to the option bytes, the hardware will first erase the whole page corresponding to the option byte, and then write to the option bytes with the value of FLASH_OPTR or FLASH_WRPR register. Furthermore, the hardware automatically calculates the corresponding complement and writes the calculated value to the corresponding area of the option bytes.

Option byte loading

After the BSY bit has been cleared, all new option bytes are written to the flash information memory but are not applied to the chip system. A read operation of the option bytes register still returns the values from the last loaded option bytes. Only when they (the new values) have been loaded does it work for the chip system.

Option bytes are loaded in the following two cases:

1) when the OBL_LAUNCH bit in the FLASH_CR register is set
2) after a power-on reset (POR)

The "load option bytes" operation is performed by reading the option bytes in the information memory area and storing the read data in the internal option registers (FLASH_OPTR and FLASH_WRPR). These internal registers configure the system and can be read by software. Setting the OBL_LAUNCH bit generates a reset so that the option bytes can be loaded with the system reset.

Each option bit has a corresponding complement at its same double word address (next half word). During the option bytes loading, the option bit and its complement are verified to ensure that the loading has been carried out correctly.

If the positive complement matches, the option bytes are copied into the option register.

If the positive complement does not match, the OPTVERR status bit of the FLASH_SR register is set. The unmatched value is written to the option register:

- for user option
  - RDP bit is written to 0xff (i.e. level 1)
  - The rest of the unmatched values are written to 1
- For the WRP option, the unmatched value is the default value of "unprotected"

After a system reset, the contents of the option bytes are copied to the following option registers (software readable and writable):

3.2.8. Flash configuration bytes

A part of the information area of the flash in the chip is used as a Factory byte to store the TRIM, CP, FT, etc. information of the chip.

3.2.8.1. Flash memory protection

The protection of the Flash main memory consists of the following mechanisms:

- read protection (RDP) to prevent accesses from external sources.

- wrtie protection (WRP) control to prevent unwanted write operations (due to confusion of the program memory pointer PC). The granularity of the write protection is designed to be 32Kbytes.
- Option byte write protection, specially designed for unlocking.

3.2.8.2. Flash memory protection

Read protection can be activated by setting the RDP option byte and performing a system reset (POR or OBL reset) to load a new RDP option byte. rdp protects flah main memory, backup registers.

If the read protection is set while the debug via SWD is still connected, a power-on reset is required instead of a system reset.

Flash memory is protected when the RDP option byte and the complement pair are correctly present in the option byte.

Table 3-5 flash memory read protection status

RDP byte value	RDP complemented byte value	Read protection level
0xAA	0x55	Level 0
Any value other than the combination [0xAA,0x55]		Level 1

System memory is readable regardless of the level of protection, but cannot be programmed or erased.

Level 0: no protection

Read, program and erase operations on the main flash are possible, as are any operations on the option byte.

Level 1: Read protection

Level 1 read protection is in effect when the RDP and its complement in the option byte contain any combination other than [0xAA,0x55], Level 1 is the default level of protection.

- User mode: program executed in user mode (boot from main flash), all operations can be performed on main flash, option byte.
- Debug, boot from SRAM, and boot from system memory modes: In debug mode, or when booting from SRAM or system memory, main flash is not accessible. In these modes, read or write accesses to main flash generate a bus error, and a hard fault interrupt.

Changing the Read protection level

The read protection level can be changed:

- from Level 0 to Level 1, changing the value of the RDP byte to any value other than 0xAA
- from Level 1 to Level 0, changing the value of the RDP byte to 0xAA

Level 1 to Level 0 will trigger a hardware mass erase Main Flash.

RDP protection summary

To be Accessed Area	READ Protection level	Boot or executed From Main Flash			Debug/ Boot or executed From SRAM/ Boot or executed From System memory			DMA		
		Read	Write	Erase	Read	Write	Erase	Read	Write	Erase
Main Flash	0	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A	N/A
	1	Yes	Yes	Yes	No	No	No	No	N/A	N/A
Information Block	x	Yes	No	No	Yes	No	No	Yes	N/A	N/A
		Yes	No	No	Yes	No	No	Yes	N/A	N/A
Option bytes (registers)	x	Yes	Yes	-	Yes	Yes	-	Yes	Yes	-
		Yes	Yes	-	Yes	Yes	-	Yes	Yes	-

Figure 3-2 Access status with protection level and execution mode

Notes:

1) A mass erase command initiated by any zone will erase Main Flash.
2) Information Block is only read-accessible, regardless of the protection level and execution mode.
3) RDP changes from Level 1 to Level 0 will trigger a hardware mass erase on the main flash.
4) For executing programs from SRAM or system memory there are two cases:

One is Boot from, the other is boot from another memory and the program jumps to SRAM or system memory for execution.

3.2.8.3. Flash Write Protection (WRP)

Flash can be set to write protect against unwanted write operations. The WRP register is defined to have a control granularity of 32Kbytes per bit of the Write Protected (WRP) area, i.e. 1 block size. See the description of the WRP register for details.

When a WRP area is activated, erase or program operations are not allowed. Accordingly, the mass erase function does not work even if only one area is set to write protect.

In addition, if an erase or program operation is attempted on a write-protected area, the write-protected error flag (WRPERR) in the FLASH_SR register will be set.

Note: Write protection only works for main flash, not for system memory.

3.2.9. Flash memory interrupts

Table 3-6 FLASH interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Enable control bit
End of operation	EOP	Write EOP=1	EOPIE
Write protection	WRPERR	Write WRPERR=1	ERRIE

The following events do not have a separate interrupt flag, but generate a Hard fault:

- Sequence error for FLASH_CR register of Unlock flash memory
- Sequence error for Unlock flash option bytes write operation
- FLASH program,erase operation not aligned for 32-bit data
- Write operation to option byte register not aligned to 32-bit data

3.3. Register description (base address 0x4002_2000)

3.3.1. FLASH access control register(FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0700

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

Bit	Name	R/W	Reset Value	Function
31:4	Reserved	RES	-	Reserved
3:0	LATECY[3:0]	RW	0	Flash read operation corresponds to the wait state: 0: Flash read operation has no wait state 1: flash read operation has 1 wait state 3: Flash read operation has 3 wait states 4: Flash read operation has 4 wait states 5: Flash read operation has 5 wait states 6: Flash read operation has 6 wait states Others：reserved Note: When configuring the latecy bit, no other bits in this register can be written to any value

3.3.2. FLASH key register(FLASH_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

All register bits are write-only and read-out returns 0.

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

Bit	Name	R/W	Reset Value	Function
31:0	KEY[31:0]	W	0x0000	The following values must be written consecutively to unlock the

				FLASH_CR register and enable the program/erase operation of the flash KEY1: 0x4567 0123 KEY2: 0xCDEF 89AB

3.3.3. FLASH option key register(FLASH_OPTKEYR)

Address offset:0x0C

Reset value:0x0000 0000

All register bits are write-only and read-out returns 0.

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

Bit	Name	R/W	Reset Value	Function
31:0	OPTKEY[31:0]	W	0x0000	The following values must be written consecutively to unlock the option register of the flash and enable program/erase operation of the option byte KEY1: 0x0819 2A3B KEY2: 0x4C5D 6E7F

3.3.4. FLASH status register(FLASH_SR)

Address offset:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSY
															R
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OPTVERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPERR				EOP
RC_W1											RC_W1				RC_W1

Bit	Name	R/W	Reset Value	Function
31:17	Reserved	RES	-	Reserved
16	BSY	R	0	Busy bit

Bit	Name	R/W	Reset Value	Function
				This bit indicates that a flash operation is in progress. This bit is set by hardware at the beginning of a flash operation and is cleared by hardware when the operation is completed or when an error is generated.
15	OPTVERR	RC_W1	0	Option and trimming bits loading validity error When the option and trimming bit and its inverse code do not match, the hardware sets the bit. Loading of mismatched option bytes is forced to a safe value, refer to section FLASH option byte program-ming. Software write 1 to clear zero.
14:5	Reserved	RES	-	Reserved
4	WRPERR	RC_W1	0	Write protection error When the address to be programmed/erased is in a write protected flash area (WRP), the hardware sets this bit. Write 1, clear the bit.
3:1	Reserved	RES	-	Reserved
0	EOP	RC_W1	0	Hardware set when the program/erase operation of flash completes successfully. This bit will only be set if the EOPIE bit in the FLASH_CR register is enabled. Write 1 to clear this bit.

3.3.5. FLASH control register(FLASH_CR)

Address offset:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	PGSTRT	Res.	OPTSTRT	Res.
RS	RS			RC_W1		RW	RW					RW		RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	BER	SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MER	PER	PG
			RW	RW									RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Lock	RS	0	FLASH_CR Lock bit. This bit can only be set by software. When set, the FLASH_CR register is Locked. When the unlock timing is successfully given, the bit is cleared by hardware and the

Bit	Name	R/W	Reset Value	Function
				FLASH_CR register is unlocked. [Software has to set this bit after the program/erase operation is completed] When an unsuccessful unlock timing is given, the bit remains set until the next system reset.
30	OPTLOCK	RS	0	Option bytes Lock bit. This bit can only be set by software. When set, the bit in the FLASH_CR register associated with option bytes is locked. When the unlock timing is successfully given, this bit is cleared by hardware and the FLASH_CR register is unlocked. The software has to set this bit after the program/erase operation is completed] When an unsuccessful unlock timing is given, the bit remains set until the next system reset.
29：28	Reserved		-	
27	OBL_LAUNCH	RC_W1	0	Force the option bytes loading。 When set, this bit forces the system to perform a reload of the option bytes. This bit is only cleared by hardware when the option byte load has been completed. If the OPTLOCK bit is set, this bit cannot be written. 0: Option byte loading complete 1: Option byte loading request is generated and the system generates a reset for option byte reloading.
26	Reserved		-	
25	ERRIE	RW	0	Error interrupt enable bit, when WRPERR bit of FLASH_SR register is set, if this bit is enabled, interrupt request will be generated. 0: No interrupt is generated 1: Interrupt generated
24	EOPIE	RW	0	End of operation interrupt enable When the EOP bit in the FLASH_SR register is set, this bit enables the generation of interrupts. 0: EOP interrupt off 1: EOP interrupt enable
23：18	Reserved	RW	0	
19	PGSTRT	RW	0	The start bit of the program operation of Flash main memory. This bit starts the program operation of Flash main memory, is set by software and is cleared by hardware

Bit	Name	R/W	Reset Value	Function
				after the BSY bit of FLASH_SR register is cleared.
18	Reserved		-	
17	OPTSTRT	RW	0	Flash option bytes modification initiation bit This bit initiates the modification of option bytes. It is set by software and cleared by hardware after the BSY bit of the FLASH_SR register has been cleared. Note: When modifications are made to the flash option bytes, the hardware automatically erases the entire 128Bytes page and then performs a program operation, which also includes automatic complement writing.
16:13	Reserved		-	
12	BER	RW	0	Block erase operation 0: Block erase operation without flash selected 1: Block erase operation with flash selected
11	SER	RW	0	Sector erase operation 0: Sector erase operation without flash selected 1: Sector erase operation with flash selected
10：3	Reserved		-	
2	MER	RW	0	Mass erase operation 0: Mass erase operation without flash selected 1: Mass erase operation with flash selected
1	PER	RW	0	Page erase operation 0: page erase operation without flash selected 1: page erase operation with flash selected
0	PG	RW	0	Program operation 0: program operation without flash selected 1: Program operation with flash selected

3.3.6. FLASH option register(FLASH_OPTR)

Address offset:0x20

Reset value: 0x0000 XXXX。

After the power-on reset (POR/OBL_LAUNCH), the corresponding value is read from the option bytes area of the flash information memory and written to the corresponding option bit of this register.

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

Res.	nRST STDBY	nRST STOP	IWDG _SW	Res.			Res.	RDP[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			
15	Reserved			
14	nRST_STDBY	RW		0： Reset generated when entering Standby mode 1： No reset generated
13	nRST_STOP	RW		0： Reset generated when entering Stop mode 1： No reset generated
12	IWDG_SW	RW		0：Hardware watchdog 1：Softwarewatchdog
11:8	Reserved	RW		
7:0	RDP	RW		0xAA：level 0, read protection Invalid Non 0xAA：level 1, read protection Valid

3.3.7. FLASH WRP register(FLASH_WRPR)

Address offset:0x2C

Reset value: 0x0000 XXXX。

After a power-on reset (POR/OBL_LAUNCH), the corresponding value is read from the option bytes area of the flash information memory and written to the corresponding value in this register.

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

Bit	Name	R/W	Reset Value	Function
31：12	Reserved			
11：0	WRP[11:0]	RW		Write protection，and each Bit corresponds to a block

				0：block N，Write protected 1：block N，Invalid write protection N=0-11

3.3.8. FLASH sleep time config register(FLASH_STCR)

Address offset:0x90

Reset value: 0x0000 6400

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			
15：8	SLEEP_TIME[7:0]	RW	0x64	FLASH sleep time configuration(counter based on HSI_10M) When LSI or LSE is selected for the system clock, this register can be used to obtain a more optimal Run mode power consumption. When LSI or LSE is selected as the system clock, the function of this register can be used for better Run mode power consumption. (recommended only when LSI or LSE is the system clock). (recommended only when LSI or LSE is the system clock). When this function is enabled, the Flash will be in Sleep every half system clock low cycle. When this function is enabled, the width of time that the Flash is in Sleep for each half system clock low cycle is tHSI_10M * SLEEP_TIME Note. tHSI_10M is the period of HSI_10M; To ensure correct Flash function, the maximum setting of this register is recommended to be 0x28. This register is recommended to be set to 0x28 in order to

Bit	Name	R/W	Reset Value	Function
				ensure correct Flash functionality. In addition, the setting value of this register is stored in 0x1FFF XXX at the factory to improve usability.
7：1	Reserved			
0	SLEEP_EN	RW	0x0	FLASH Sleep enable 1： enable flash sleep 0： disable flash sleep

3.3.9. FLASH TS0 register(FLASH_TS0)

Address offset:0x100

Reset value: 0x0000 003C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS0							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			
7：0	TS0	R	0x3C	TS0 = 7.5*Freq_HSI8M

3.3.10. FLASH TS1 register(FLASH_TS1)

Address offset: 0x104

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			
7：0	TS1	R	0x90	TS1 = 18*Freq_HSI8M

3.3.11. FLASH TS2P register(FLASH_TS2P)

Address offset: 0x108

Reset value: 0x0000 003C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS2P							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			
7：0	TS2P	R	0x3C	TS2P = 7.5*Freq_HSI8M

3.3.12. FLASH TPS3 register(FLASH_TPS3)

Address offset: 0x10C

Reset value: 0x0000 0240

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	TPS3										
					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：11	Reserved			
10：0	TPS3	R	0x240	TPS3 = 72*Freq_HSI8M

3.3.13. FLASH TS3 register(FLASH_TS3)

Address offset: 0x110

Reset value: 0x0000 003C

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：8	Reserved			
7：0	TS1	R	0x3C	TS1 = 7.5*Freq_HSI8M

3.3.14. FLASH ERASE TPE register(FLASH_ERSTPE)

Address offset: 0x114

Reset value: 0x0000 4E20

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			
15：0	ERSTPE	R	0x5DC0	ERSTPE = 3000*Freq_HSI8M

3.3.15. FLASH PROGRAM TPE register(FLASH_PRGTPE)

Address offset: 0x118

Reset value: 0x0000 2EE0

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved			
15：0	PRGTPE	R	0x1F40	PRGTPE = 1000*Freq_HSI8M

3.3.16. FLASH PRE-PROGRAM TPE register (FLASH_PRETPE)

Address offset: 0x11C

Reset value: 0x0000 0640

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	PRETPE												
			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：13	Reserved			
12：0	PRETPE	R	0x640	PRETPE = 200*Freq_HSI8M

3.3.17. FLASH register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FLASH_ACR	Reserved																												LATENCY			
	Read/Write																													RW			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0x08	FLASH_	KEY																															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	KEYR																																
	Read/Write																	W															
	ResetValue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	FLASH_OPTKEYR																																
	Read/Write																	W															
	ResetValue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	FLASH	Reserved															BSY	OPTVE	Reserved									WRPE	Reserved			EOP

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	_SR																	RR											RR				
	Read/Write																R	RC_W1											RC_W1				RC_W1
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	FLASH_CR	LOCK	OPTLOCK	Reserved		OBL_LAUNCH	Reserved	ERRIE	EOPIE	Reserved				PGSTRT	Reserved	OPTSTRT	Reserved	Reserved			BER	SER	Reserved						MER	PER	PG		
	Read/Write	RS	RS			RC_W1		RW	RW					RW		RW					RW	RW							RW	RW	RW		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	FLASH_OP	Reserved																Nboot1	NRST_MODE	WWDG_SW	IWDG_SW	Reserved				RDP							

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TR																																
	Read/Write																	RW	RW	RW	RW					RW	RW	RW	RW	RW	RW	RW	RW
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x2C	FLASH_WRPR	Reserved																Reserved				WRP											
	Read/Write																					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X

4. Power control (PWR)

4.1. Functional description

4.1.1. Power supply structure

The power supply structure is shown below：

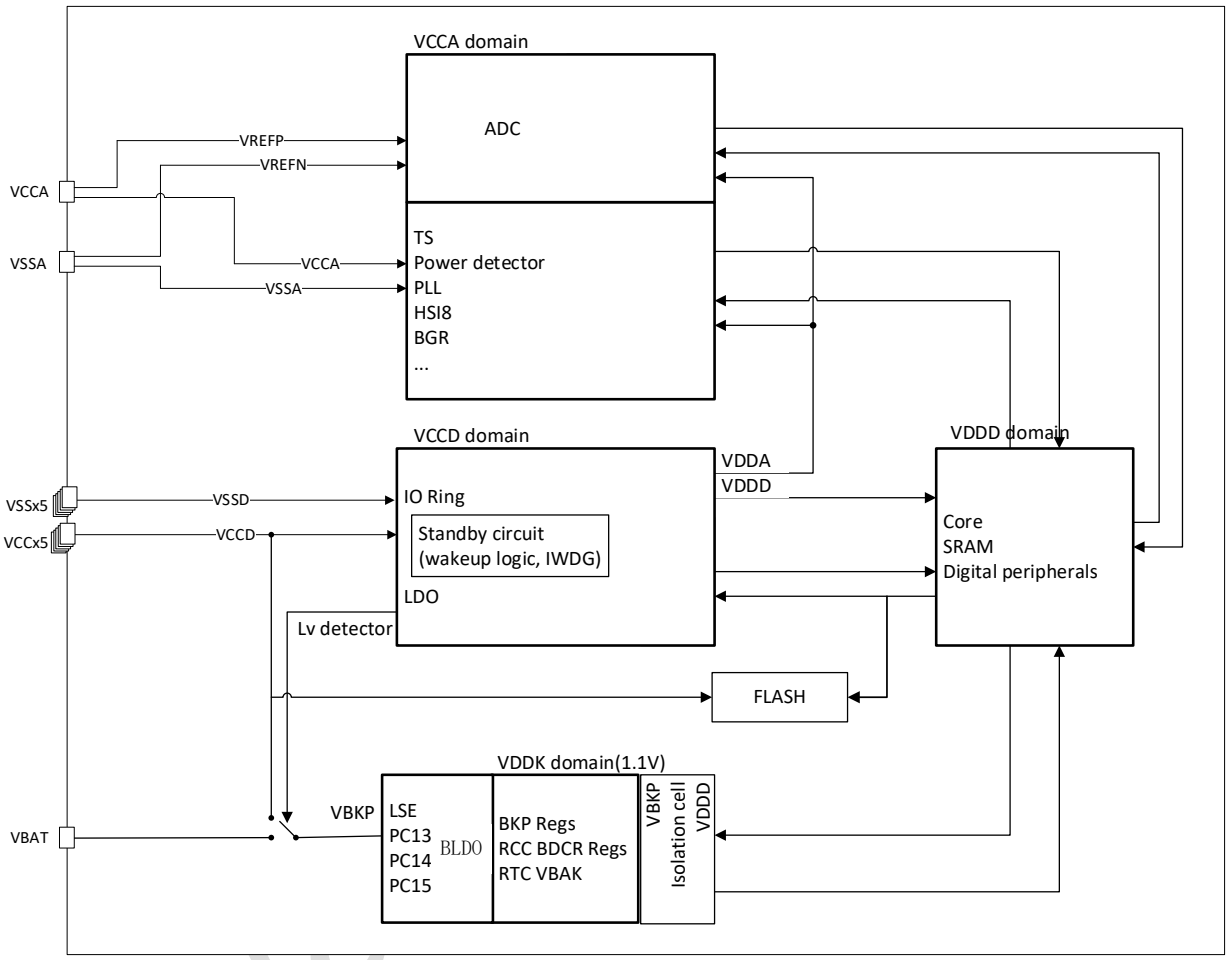


Figure 4-1 Power supply structure

Table 4-1 Power supply structure

No.	Power supply	Power value	Description
1	VCC	1.7V~3.6V	Power is provided to the chip via the power supply pins. The power supply modules are: part of the analogue IP and IO circuitry, and the standby mode control logic.
2	VCCA	1.7V~3.6V	Powering most of the analogue modules and most of the IOs from the VCCA PAD.
4	VDDx (VDDD/VDDA)	1.1V/1.0V/0.9V/0.8V± 10%	The output from the VR supplies the main logic circuits inside the chip, the SRAM. When MR is powered, the output is 1.1v. When in stop mode, it can be powered by MR or LPR depending on the software configuration, and the LPR output is 1.1V/1.0V/0.9V/0.8V depending on the software configuration.

4.1.1.1. Main power supply for the system

Independent A/D converter power supply and reference voltage

To improve conversion accuracy, the ADC is equipped with a separate power supply that can be filtered separately and shielded from noise on the PCB.

- The ADC voltage source is fed from a separate VCCA pin.
- The VSSA pin provides a separate power supply ground connection.

To improve the accuracy of the low voltage input, the user can connect a separate ADC external reference voltage input to VREF. The voltage range on VREF is 1.7 V to VDDA.

Backing up the battery field

To retain the contents of the RTC backup register and power the RTC after a VCCD power failure, the VBAT pin can be connected to a battery or other backup power supply.

For the RTC to work even after the main digital power supply (VDD) has been switched off, the VBAT pin needs to be powered for each of the following modules:

- RCC BDCR register
- RTC module (except RTC_CR register)
- LSE oscillator
- BKP registers
- PC13 to PC15 I/O

The switching of the VBAT power supply is controlled by the built-in power-down reset circuit (BPOR) in the reset module.

Note:

- The power switch between VBAT and VCCD remains connected to VBAT after tRSTTEMPO (temporisation time at VCC start-up) or after PDR is detected.
- During the start-up phase, if the established VCCD is less than tRSTTEMPO (see tRSTTEMPO value in the data sheet) and VCCD > VBAT + 0.6 V, current can be injected into VBAT via the internal diode connected between VDD and the power switch (VBAT).

If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low drop diode between this power supply and the VBAT pin.

If no external battery is used in the application, it is recommended to connect the VBAT pin to the VCCD with a 100 nF decoupling ceramic capacitor connected in parallel to the VCCD.

When powering the backup domain via the VCCD (analogue switch connected to the VCCD) the following pins can be implemented:

- PC14 and PC15 can be used as GPIO or LSE pins
- PC13 can be used as GPIO and can also be configured for other functions (TAMPER pin, RTC calibration clock, RTC alarm or seconds output)

Note: Due to the limited current fill capability of this switch (3 mA), the following limitations exist when using GPIOs PC13 to PC15 in output mode: the rate must not exceed 2 MHz, the maximum load is 30 pF and these I/Os cannot be used as current sources (e.g. for driving LEDs).

When powering the backup domain via VBAT (as there is no VCCD, the internal power switch is connected to VBAT), the following pinouts are possible:

- PC14 and PC15 can only be used as LSE pins

- PC13 can be used as RTC additional function pin (TAMPER pin, RTC alarm or seconds output)

Backup Domain Access

After reset, the backup domains (RTC registers and RTC backup registers) are protected against accidental write access. To enable access to the backup domain (RTC and RTC backup registers), configure as follows:

- Enable the power interface clock by placing PWREN position 1 in the RCC_APB1ENR register
- Enable access to the backup domain by placing DBP in the PWR_CR register in position 1
- Configure the RTCSEL register in the RCC Backup Domain Control Register (RCC_BDCR) to select the RTC clock source
- Configure RTCEN in the RCC Backup Domain Control Register (RCC_BDCR) to enable RTC clocking

VDDD Regulators

The embedded linear VR powers all digital circuits other than the backup domain and standby (wake-up) circuits. The regulator output voltage defaults to 1.1 V and can be configured to output 1.0 V, 0.9 V, and 0.8 V. The regulator output voltage is 1.1 V by default.

When activated via software, the regulator is always enabled after a reset. Depending on the application mode, three different modes of operation are available:

- In run mode, the regulator provides full power to the 1.1 V domain (core, memory and digital peripherals). In this mode, the regulator output voltage can be adjusted by software to different voltage values (configurable via the VOS[1:0] bits of the PWR_CR register).
- In stop mode, the main or low-power regulator provides a low-power voltage to the VDDD domain to save the contents of the registers and internal SRAM. It is also possible to place the regulator in master regulator mode (MR) or low power mode (LPR).
- In standby mode, the regulator is powered down. The contents of the registers and SRAM are lost, except for the standby (wake-up) circuitry and the backup field.

VBKP Regulators

The embedded linear VR powers all digital circuits in the backup domain. The regulator output voltage defaults to 1.1 V and can be configured to output 0.9 V, 0.85 V, and 0.8 V. The regulator output voltage is 1.1 V by default.

When activated via software, the regulator is always enabled after reset. Depending on the application mode, three different modes of operation are available:

- In run mode and low power mode, the regulator provides full power to the VDDK domain;
- operates always in all modes;

4.1.1.2. power detection

POR/PDR

The chip contains power-on reset (POR) and power-off reset (PDR) modules to provide power-related reset. Among them, POR/PDR works by default in all power modes.POR/PDR output reset low is active.

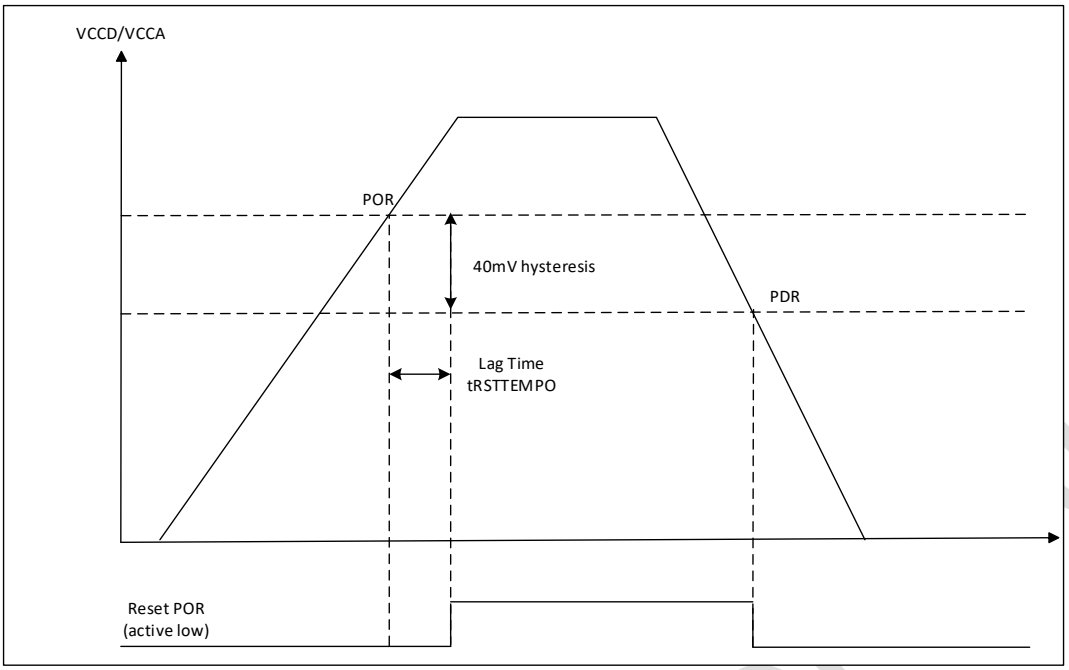


Figure 4-2 POR/PDR Timing

PVD Module

The PVD module is to detect whether VCC is below the threshold set by PWR_CR.PLS.

The PVD function is enabled by configuring the PVDE register.

The PVDO flag in the Power Control/Status Register (PWR_CSR) is used to indicate whether VCC is above or below the voltage threshold for PVD. This event is internally connected to Line16 of the EXTI and generates an interrupt if the interrupt is enabled in the external interrupt register. The PVD interrupt is generated when VCC falls below the PVD threshold and/or when VCC rises above the PVD threshold, depending on the rising/falling edge trigger setting of EXTI Line16. In practice, this feature can be used for use in performing emergency shutdown tasks.

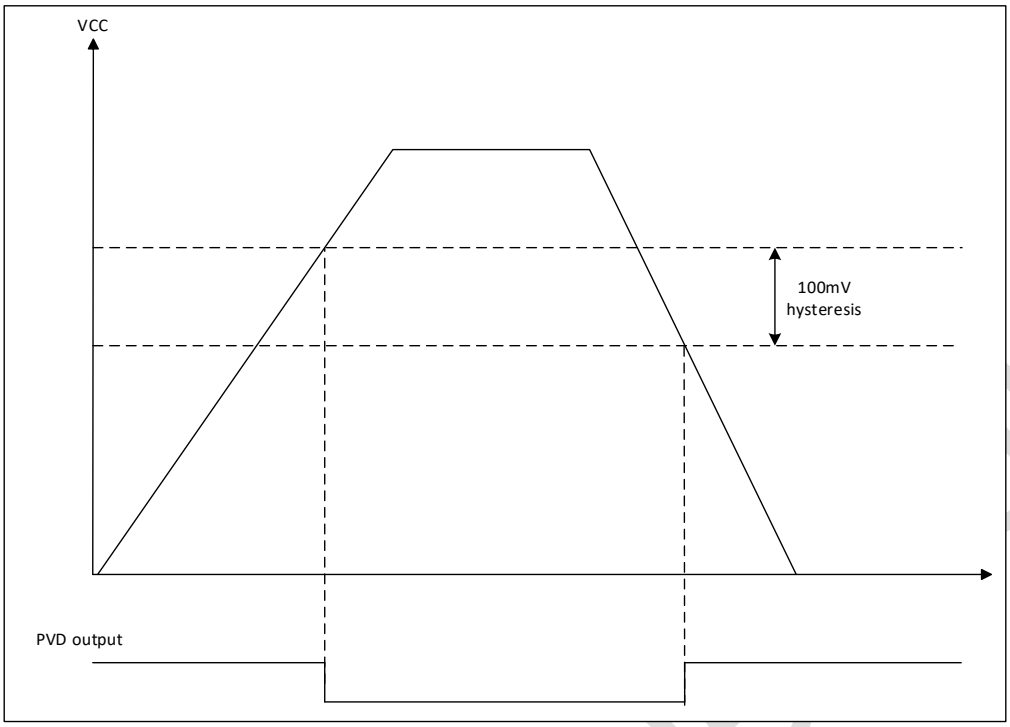


Figure 4-3 PVD Timing

4.1.2. System Low Power Mode

By default, the microcontroller enters run mode after a system reset or power-on reset. In run mode, the CPU provides a clock via HCLK and executes program code. The system provides several low-power modes to save power consumption when the CPU does not need to run (e.g., when waiting for external events). It is up to the user to select the specific low-power mode according to the application, seeking the best balance between low power consumption, short boot time, and available wake-up sources.

The device has three low-power modes:

- Sleep mode (Cortex®-M4F core with FPU is stopped, peripherals including those of the Cortex®-M4 (NVIC, Sys Tick) etc. can be configured to run)
- Stop mode (HSI48M, HSI8M, HSE and PLL clocks are stopped)
- Standby mode (VDDD domain powered down)

VBAT power is present on the chip, so when VCC is powered down, the chip operates only in the VBKP domain.

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

- Reducing the system clock speed
- Turn off the corresponding peripheral clocks when APBx and AHBx peripherals are not used.

4.1.2.1. Enter Low Power Mode

The MCU enters low-power mode by executing the WFI (wait for interrupt) or WFE (wait for event) instruction, or by setting SLEEPONEXIT in the Cortex®-M4F system control register with FPU to position 1 when returning from the ISR.

Low power mode can be entered via WFI or WFE only when there are no interrupts or events pending, i.e., the CPU needs to clear pending interrupts or events before entering low power.

If a wake-up signal occurs during the process of entering low-power mode, the low-power entry process is exited.

4.1.2.2. Exit low-power mode

The MCU exits both low power modes depending on how it entered sleep and stop mode:

- If low power mode is entered using the WFI instruction or returning from the ISR, any peripheral interrupt acknowledged through the NVIC can wake up the device.
- If the WFE instruction is used to enter low-power mode, the MCU will exit low-power mode as soon as an event occurs. The wake-up event can be generated in the following ways:

✓ NVIC IRQ interrupt:

Enable the interrupt in the peripheral control register, not in the NVIC, and enable SEVONPEND=1 in the Cortex®-M4F system control register. when the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral's NVIC interrupt channel pending bit (in the NVIC interrupt clear pending register) must be cleared.

✓ event:

Configure an external or internal EXTI line for event mode. When the CPU recovers from WFE, it is not necessary to clear the interrupt pending bit of the EXTI peripheral or the NVIC IRQ channel pending bit because the pending bit of the corresponding event line is not set to one. It may be necessary to clear the interrupt flag in the peripheral. This mode takes the shortest amount of time to wake up because there is no time to spend on interrupt entry or exit.

The MCU exits standby low-power mode when an external reset (NRST pin) occurs, an IWDG reset occurs, a rising edge occurs on the enabled WKUPx pin, or an RTC wake-up event is triggered. After waking up from standby mode, the programme will be re-executed as it was after the reset (boot pin sampling, reset vector fetch, etc.).

4.1.2.3. Low Power Mode Summary

Table 4-2 Low Power Mode Summary

mode	entry condition	exit condition	wake-up delay	VDDD domain state	VCCD domain state	VDDK domain state	IO state	else
Sleep mode 1、SLEEPONEXIT=0 时 Executing WFI or WFE command, the CPU enters the sleep mode immediately; 2,	Perform WFI or return from ISR and SLEEPING=1, SLEEPDEEP=0	Any interruption	no delay	Processor core clock off; Other clocks switched on and off according to configu	Standby mode wake-up module does not work; Whether the IWDG works	Normal operation; Whether LSE/RTC/BKP is working is configured by the CPU; TrimReg module working;	Maintains original operating mode status	NA
	Execute WFE with SLEEPING=1, SLEEPDEEP=0	Wakeup event or SEVONPEND	no delay					NA

mode	entry condition	exit condition	wake-up delay	VDDD domain state	VCCD domain state	VDDK domain state	IO state	else
SLEEPONEXIT = 1 when the system exits from the lowest priority interrupt handler, the CPU immediately enters the sleep mode.		interrupt when 1		ration;	or not is configured by the CPU;			
Stop Mode	Execute WFE or WFI or return from ISR when the following conditions are met: 1, SLEEPDEEP=1; 2, PDDS=0	Interrupt or wake-up event generation WFI entry: any EXTI line configured for wake-up from a stop-mode interrupt; WFE entry: any EXTI line configured to wake up from a stop mode event; IWDG reset; NRST reset;	HSI8M wake-up time + (LDO exit time from low-power mode) + FLASH wake-up time (time may overlap depending on configuration)	Turn off HSI8M, HSI48M, PLL, HSE; SRAM enters retention; Flash into sleep; LPDS=0: normal open; LPDS=1: low power mode open	Standby mode wake-up module does not work; Whether the IWDG works or not is configured by the CPU;	Normal operation; Whether LSE/RTC/BKP is working is configured by the CPU; TrimReg module working;	Maintain the original operating mode status	IWDG, RTC, LSI, LSE32K need to be enabled or disabled respectively before entering stop mode if required.
standby mode	Execute WFE or WFI or return from ISR when the following conditions are met:	Rising edge of the WKUP pin (when enabled); Rising edge of the RTC alarm event;	Includes the reset process initiated by the LDO/HSI8M 01	Power down; Loss of SRAM contents;	The standby wake-up module	Normal operation; Whether the LSE/RTC works or not is	Highly resistive state except for	IWDG, RTC, LSE32K can be set independently

mode	entry condition	exit condition	wake-up delay	VDDD domain state	VCCD domain state	VDDK domain state	IO state	else
	1.SLEEPDEEP=1; 2. PDDS=1; 3. Clear the WUF bit in the power control/status register (PWR_CSR)	external reset on the NRST pin; IWDG reset;			works normally; Whether the IWDG works or not is configured by the CPU;	configured by the CPU;	the following IO pins: 1. NRST 2. Intrusion detection or RTC calibration output PIN 3. IO pin corresponding to WKUP function 4. LSE related IO	as required

All interrupt pending bits must be cleared before entering sleep mode.

4.1.2.4. Sleep mode

Table 4-3 Sleep Modes

Sleep mode	Descriptions
enter mode	WFI or WFE and: --SLEEPDEEP=0 and

Sleep mode	Descriptions
	--SLEEPONEXIT=1 and --No interrupt hangs.
	Returns from the lowest priority ISR and: --SLEEPDEEP=0 and --SLEEPONEXIT=1 .
exit mode	Use WFI or return from lowest priority ISR to enter: interrupt (enable)
	Entered with WFE and SEVONPEN=0: Wake-up event
	Entered with WFE and SEVONPEN=1: peripheral enable interrupt (even if disabled in NVIC)
wake-up delay	NA

4.1.2.5. Stop Mode

Stop mode is based on the Cortex®-M4F SleepDeep mode with FPU with peripheral clock gating.The VDDD domain LDO can be configured either in normal mode or in low power mode. In stop mode, all high frequency clocks in the VDDD domain are stopped. the stopped clocks include PLL, HSI8M, HSI48M, and HSE in the order of PLL->HSI48->HSE->HSI8. the internal SRAM and register contents will be retained.

Certain settings in the PWR_CR register can further reduce power consumption. waking up the device from stop mode when Flash is in stop mode requires an additional startup delay.

If Flash programming is being performed, stop mode entry is delayed until the end of the memory access (controlled by the HREADY signal at the FLASH controller interface and the operation flag, software can execute the WFI or WFE instruction after the end).

If the APB domain is being accessed before entering stop mode, stop mode entry will be delayed until the end of the APB access. (Guaranteed by software)

In stop mode, the following functions can be selected by programming each control bit:

- Independent Watchdog (IWDG): The IWDG is started by writing to its key register or by using the hardware option. Once started, it cannot be stopped except by a reset.
- Real Time Clock (RTC): Configured via the RTCEN bit in the RCC Backup Domain Control Register (RCC_BDCR).
- Internal RC Oscillator (LSI RC): configured via the LSION bit in the RCC Clock Control and Status Register (RCC_CSR).
- External 32.768 kHz Oscillator (LSE OSC): configured via the LSEON bit in the RCC Backup Domain Control Register (RCC_BDCR).
- PVD: configured via the PVDE bit in the Power Control Register PWR_CR.

Exiting stop mode will select the HSI8M as the system clock.

Table 4-4 Stop mode

Stop mode	Description
enter mode	WFI or WFE, and: --No interrupt (for WFI) or event (for WFE) is pending; --SLEEPDEEP=1; --Clear the PDDS bit in the PWR_CR register to zero;

Stop mode	Description
	--Configure the LPDS in the PWR_CR register to select the VR operating mode;
	Return from the lowest priority ISR and: --SLEEPDEEP=1; --SLEEPONEXIT=1; --No interrupt pending; --Clear the PDDS bit in the PWR_CR register to zero; --Configure the LPDS in the PWR_CR register to select the VR operating mode;
	Note: To enter stop mode, all EXTI line pending bits (EXTI_PR), all peripheral interrupt pending bits, and the RTC alarm flag bit must be reset. Otherwise entry into stop mode will be ignored and program execution will continue (CPU implementation).
exit mode	Use WFI or return from ISR to enter: any EXTI line configured for interrupt mode (while enabling the corresponding EXTI interrupt in the NVIC). The interrupt source is an external interrupt or an interrupt generated by a peripheral with a wake-up function.
	Entered with WFE and SEVONPEND=0: Any EXTI line configured for event mode.
	Entered with WFE and SEVONPEND=1: - - Any EXTI line configured for interrupt mode (even if the EXTI interrupt is disabled in the NVIC interrupt counterpart). The interrupt source is an external interrupt or an interrupt generated by a peripheral with wake-up capability. --Wake-up events
	NRST reset
	IWDG reset
wake-up delay	HSI8M wake-up time + (LDO exit time from low-power mode) + FLASH wake-up time (time may overlap depending on configuration)

4.1.2.6. Standby mode

In standby mode, the VR stops working and the VDDD domain is powered down. the PLL, HSI8M, HSI48M, and HSE are all turned off, and the order of shutdown is PLL->HSI48->HSE->HSI8. the SRAM and register contents are lost except for the registers in the VBAK domain (the RTC registers and the backup register) and the standby circuit.

In standby mode, the following functions can be selected by programming each control bit:

- Independent Watchdog (IWDG): The IWDG is activated by writing to its key register or using a hardware option. Once started, it cannot be stopped except by a reset.
- Real Time Clock (RTC): Configured via the RTCEN bit in the RCC Backup Domain Control Register (RCC_BDCR).
- Internal RC Oscillator (LSI RC): configured via the LSION bit in the RCC Clock Control and Status Register (RCC_CSR).
- External 32.768 kHz Oscillator (LSE OSC): configured via the LSEON bit in the RCC Backup Domain Control Register (RCC_BDCR).

Table 4-5 Standby mode

Standby mode	description
enter mode	WFI or WFE, and: --No interrupt (for WFI) or event (for WFE) is pending; --SLEEPDEEP=1; --Position the PDDS bit in the PWR_CR register; --Configure the WUF bit in the PWR_CSR register to be cleared to zero (if not, it will wake up immediately); --Clear the flag corresponding to the selection of the wake-up source;
	Return from ISR and: --SLEEPDEEP=1; --SLEEPONEXIT=1; --No interrupt pending; --Position the PDDS bit in the PWR_CR register; --Clear the WUF bit in the PWR_CSR register to zero (if not cleared, it will wake up immediately); --Clear the flag corresponding to the selection of the wake-up source;
exit mode	WKUPx pin (x=1, 2, 3, 4, 5)
	RTC wake-up event generation
	NRST pin reset
	IWDG reset
wake-up delay	HSI8M wake-up time + LDO power-up time + LDO wait time (configurable 5us/10us/20us/30us) + FLASH wait time 3us (depending on the configuration time can have overlap)

In standby mode, all IOs are in the high resistance state except for the following components:

- Reset pin
- TAMPER pin when set to anti-intrusion or calibration output (TBD)
- 5 WKUP pins (PA0/PC13/PE6/PA2/PC5) (if enabled)

4.1.2.7. Debugging in stop mode and standby mode

In Stop and Standby modes, by default, the debugging function is interrupted because the CPU core is not clocked. However, if the relevant registers in DBGMCU_CR are configured, debugging is still possible even if the CPU enters SLEEPDEEP mode.

4.1.2.8. Automatic wake-up in low-power mode (AWU)

The RTC can wake up the MCU in low-power mode without relying on external interrupts (Automatic Wake-Up Mode). The RTC provides a programmable time base for periodic wake-up from stop or standby mode. Two of the three RTC clock sources can optionally be used to implement this function by programming the RTCSEL[1:0] bits of the Backup Region Control Register (RCC_BDCR):

- Low Power 32.768kHz External Crystal (LSE)

This clock source provides a low-power and accurate time reference. (Typically consumes less than 1μA).

- Low Power Internal RC Oscillator (LSI)

Using this clock source saves the cost of a 32.768kHz crystal. However, the RC oscillator will slightly increase the power consumption. In order to wake up the system from stop mode with an RTC alarm event, the following must be done:

- Configure EXTI17 for rising edge triggering.
- Configure the RTC so that it can generate wake-up events.

It is not necessary to configure EXTI17 if you want to wake up from standby mode (wake up standby directly using the RTC alarm clock wake-up event).

The registers of this peripheral can be accessed via half-word or word.

4.1.2.9. Functions in different modes

The following table shows the presence or absence of each functional module in different modes:

Table 4-6 Functions in different modes

Peripheral	Run	Sleep	Stop		Standby		VBKP/VDDK
			VR@LPR or VR@MR	Wakeup ability	-	Wakeup ability	
CPU Core	Y	-	-	-	-	-	-
Flash memory	Y	O(2)	-(3)	-	-	-	-
SRAM	Y	O(4)	-(5)	-(5)	-	-	-
PVD	O	O	O	O	-	-	-
ESMC	O	O	-	-	-	-	-
DMA	O	O	-	-	-	-	-
HSI	O	O	-	-	-	-	-
HSI48	O	O	-	-	-	-	-
HSE	O	O	-	-	-	-	-
LSI	O	O	O	-	O	-	-
LSE	O	O	O	-	O	-	O
PLL	O	O	-	-	-	-	-
HSE Clock Security System (CSS)	O	O	-	-	-	-	-
RTC/Auto Wakeup	O	O	O	O	O	O	O
Number of RTC TAMP pin	1	1	1	1	1	-	1
USB	O	O	-	-	-	-	-
USART1	O	O	-	-	-	-	-
USART2	O	O	-	-	-	-	-
USART3	O	O	-	-	-	-	-
USART4	O	O	-	-	-	-	-
USART5	O	O	-	-	-	-	-
I2C1	O	O	-	-	-	-	-
I2C2	O	O	-	-	-	-	-
SPI1	O	O	-	-	-	-	-
SPI2	O	O	-	-	-	-	-
SPI3	O	O	-	-	-	-	-
CAN	O	O	-	-	-	-	-

Peripheral	Run	Sleep	Stop		Standby		VBKP/VDDK
			VR@LPR or VR@MR	Wakeup ability	-	Wakeup ability	
ADC1	O	O	-	-			
ADC2	O	O	-	-	-	-	-
ADC3	O	O	-	-	-	-	-
Temperature sensor	O	O	-	-	-	-	-
Timers(TIM1/TIM3/ TIM6/TIM14/TIM16/TIM17)	O	O	-	-	-	-	-
IWDG	O	O	O	O	O	O	-
WWDG	O	O	-	-	-	-	-
SysTick timer	O	O	-	-	-	-	-
CRC	O	O	-	-	-	-	-
GPIOs	O	O	O	O	O	5pins	3pins
ACC	O	O	-	-	-	-	-
BKP	O(5)	O(5)	O(5)	-	O(5)	-	O(6)

Notes:

1) Y = Yes (enable); O = Optional (off by default, can be enabled by software); - = Not available
2) In sleep mode, the FLASH controller clock can be turned off.
3) In Stop mode, FLASH does not power down, but enters Sleep low power mode, wake up needs 3us wait.
4) In sleep mode, the SRAM clock can be turned on or off.
5) The SRAM does not power down, but no clock is supplied and it enters the lowest power state.
6) The BKP module clock has an enable signal defined as O

4.2. Register description (base address 0x4000_7000)

4.2.1. power control register (PWR_CR) (0x00)

Address offset: 0x00

Reset value: 0x0011 0000(Cleared on wake-up from standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	STDBY_MRRDY_WAIT			FLS_WUPT		HSION_CTRL.
										RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	VOS[1:0]		BKPVR_VOS[1:0]		Res	Res	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:22	Reserved	-	-	Reserved
21:20	STDBY_MRRDY_WAIT[1:0]	RW	2'h1	The time to wait after waking up in standby mode and MR ready.

Bit	Name	R/W	Reset Value	Function
				00：5us 01：10us 10：20us 11：30us
19	Reserved	-	-	Reserved
18:17	FLS_WUPT[1:0]	RW	2’b00	In the stop mode wake-up timing, after the HSI8M is stabilised, a waiting time is required before FLASH operation. 00：3us； 01：5us； 10：2us； 11：0us；
16	HSION_CTRL	RW	0	HSI8M turn on time control when waking up from stop mode. 0: Wait for MR to stabilise and then enable HSI; 1: turn on at the same time as VR, i.e., enable HSI immediately upon wake-up.
15	Reserved	-	-	Reserved
14:13	VOS[1:0]	RW	2’b00	Selects the VR output voltage level. This bit is used to control the output voltage of the internal VR in order to achieve a balance between device and power consumption. When the VR is in LP mode: 00: 1.1V 01: 1.0V 10: 0.9V 11: 0.8V When VR is MR mode: 0x: 1.1V 1x: 0.9V Note: This signal is not shown on the external datasheet.
12:11	BKPVR_VOS[1:0]	RW	0	PMU BLDO output voltage control. 00：VDDK=1.1V 01：VDDK=0.9V 10：VDDK=0.85V 11：VDDK=0.8V This signal is 0 when VDDD power-on reset is active
10:9	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
8	DBP	RW	0	BACKUP field register write protection. 0: RTC and RTC backup registers are not accessible; 1: RTC and RTC backup registers are accessible;
7:5	PLS[2:0]	RW	0	PVD Level Selection. Written by software to select the voltage threshold for the PVD. 000：VPVD0 (around 1.8V) 001：VPVD1 (around 2.0V) 010：VPVD2 (around 2.2V) 011：VPVD3 (around 2.4V) 100：VPVD4 (around 2.6V) 101：VPVD5 (around 2. 8V) 110：VPVD6 (around 3.0V) 111：VPVD7 (around 3.2V)
4	PVDE	RW	0	PVD enable. 0: Disable PVD; 1: enable PVD;
3	CSBF	W	0	The standby flag is cleared to zero. 0: No effect; 1: Clear the SBF flag bit to zero;
2	CWUF	W	0	Zeroes the wake-up flag. 0: No effect; 1: Clear the WUF flag to zero after 2 system clock cycles;
1	PDDS	RW	0	Power-down deep sleep. This bit is set to 1 and cleared by software. Configured in conjunction with LPDS. 0: The system enters stop mode when the CPU enters SLEEPDEEP. the VR state depends on the LPDS configuration; 1: The system enters standby mode when the CPU enters SLEEPDEEP. the VR is off;
0	LPDS	RW	0	VR Low Power Deep Sleep Configuration. This bit is set to 1 and cleared by software. Configured in conjunction with PDDS. 0: VR (MVR) on in stop mode; 1: LPVR on in stop mode;

4.2.2. power control/status registers (PWR_CSR)(0x04)

Address offset: 0x04

Reset value: 0x0000 0000(Waking up from standby mode does not reset the WUF/SBF and EWUPx bits of this register)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLT_CTRL[2:0]			FLTEN
												RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	EWUP5	EWUP4	EWUP3	EWUP2	EWUP1	Res	Res	Res	Res	Res	PVDO	SBF	WUF
			RW	RW	RW	RW	RW						R	R	R

Bit	Name	R/W	Reset Value	Function
31:20	Reserved	-	-	Reserved
19:17	FLT_CTRL[2:0]	RW	3'b000	PVD filter time configuration. 111: Reserved; 110: filter time of approximately 1024 filter clocks (approximately 30.7ms when LSI or LSE); 101: Filter time is approximately 128 filter clocks (approximately 3.8ms when LSI or LSE); 100: The filtering time is approximately 64 filter clocks (approximately 1.92ms for LSI or LSE); 011: Filtering time is approximately 16 filter clocks (approximately 480us for LSI or LSE); 010: Filtering time is approximately 4 filter clocks (approximately 120us for LSI or LSE); 001: Filtering time is approximately 2 filter clocks (approximately 60us for LSI or LSE); 000: Filtering time is approximately 1 filter clock (approximately 30us for LSI or LSE);
16	FLTEN	RW	0	PVD digital filter enable. 0: PVD digital filter disable; 1: PVD digital filter enable;
15:13	Reserved	-	-	Reserved
12	EWUP5	RW	0	Enable WKUP5 pin. This bit is set to 1 and cleared by software. 0: The WKUP5 pin is used for general purpose I/O. Events on the WKUP5 pin do not wake the device from standby mode. 1: The WKUP5 pin is used to wake the device from standby mode and is forced to be configured as an input pull-down (waking the system from standby mode when a rising edge occurs on the WKUP5 pin). Note: This bit is reset by system reset. However, this bit is not reset when waking up from standby mode.
11	EWUP4	RW	0	Enable WKUP4 pin.

Bit	Name	R/W	Reset Value	Function
				This bit is set to 1 and cleared by software. 0: The WKUP4 pin is used for general purpose I/O. Events on the WKUP4 pin do not wake the device from standby mode. 1: The WKUP4 pin is used to wake the device from standby mode and is forced to be configured as an input pull-down (waking the system from standby mode when a rising edge occurs on the WKUP4 pin). Note: This bit is reset by system reset. However, this bit is not reset when waking up from standby mode.
10	EWUP3	RW	0	Enable WKUP3 pin. This bit is set to 1 and cleared by software. 0: The WKUP3 pin is used for general purpose I/O. Events on the WKUP3 pin do not wake the device from standby mode. 1: The WKUP3 pin is used to wake the device from standby mode and is forced to be configured as an input pull-down (wakes the system from standby mode when a rising edge occurs on the WKUP3 pin). Note: This bit is reset by system reset. However, this bit is not reset when waking up from standby mode.
9	EWUP2	RW	0	Enable WKUP2 pin. This bit is set to 1 and cleared by software. 0: The WKUP2 pin is used for general purpose I/O. Events on the WKUP2 pin do not wake the device from standby mode. 1: The WKUP2 pin is used to wake the device from standby mode and is forced to be configured as an input pull-down (wakes the system from standby mode when a rising edge occurs on the WKUP2 pin). Note: This bit is reset by system reset. However, this bit is not reset when waking up from standby mode.
8	EWUP1	RW	0	Enable WKUP1 pin. This bit is set to 1 and cleared by software. 0: The WKUP pin is used for general purpose I/O. Events on the WKUP pin do not wake the device from standby mode. 1: The WKUP pin is used to wake the device from standby mode and is forced to be configured as an input pull-down (waking the system from standby mode when a rising edge occurs on the WKUP pin). Note: This bit is reset by system reset. However, this bit is not reset when waking up from standby mode.
7:3	Reserved	-	-	Reserved
2	PVDO	R	0	PVD Output Flag.

Bit	Name	R/W	Reset Value	Function
				This bit is set to 1 and cleared by hardware. This bit is valid only when PVD is enabled via the PVDE bit. 0: VDD is above the PVD threshold selected by the PLS[2:0] bits. 1: VDD is below the PVD threshold selected by the PLS[2:0] bits. Note: PVD stops when entering standby mode. Therefore, after standby mode or after reset, this bit is equal to 0 until PVDE position 1 and then the corresponding value is output according to the detected voltage threshold.
1	SBF	R	0	Standby mode flag. This bit is set to 1 by hardware, and clearing it can only be achieved by POR/PDR (VCC power-on reset/power-down reset) or by setting the CSBF in the PWR_CR register to 1. 0: The device has not entered standby mode 1: The device has entered the standby mode
0	WUF	R	0	Standby mode wake-up flag. This bit is set to 1 by hardware, and clearing it can only be done by POR/PDR (VCC power-on reset/power-down reset) or by clearing CWUF bit 1 in the PWR_CR register. 0: No wake-up event occurred 1: Wake-up event received, may be from WKUP pin, RTC wake-up event.

4.2.3. PWR Register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40007000	PWR_CR	Reserved										STDBY_MRRDY_WAIT[1:1]		Reserved	FLS_WUPT[1:0]		HSION_CTRL	Reserved	VOS[1:0]		BKPVR_VOS[1:0]		Reserved		DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Read/Write											RW			RW		RW		RW		RW				RW	RW			RW	W	W	RW	RW
	Reset V	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	alue																																
0x04	PWR_CSR	Reserved												FLT_CTRL[2:0]			FLTEN	Reserved			EWUP5	EWUP4	EWUP3	EWUP2	EWUP1	Reserved					PVDO	SBF	WUF
	Read/Write													RW			RW				RW	RW	RW	RW	RW						R	R	R
	ResetValue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1

5. Reset and clock control (RCC)

5.1. Description of the reset function

5.1.1. System reset

A system reset resets all registers to their reset values except for the reset flag in the clock control register CSR and the registers in the backup field.

A system reset is generated whenever one of the following events occurs:

- NRST pin low (external reset)
- End of window watchdog count (WWDG reset)
- End of independent watchdog count (IWDG reset)
- Software reset (SYSRESETREQ reset)
- Low power management reset (NRST_STDBY/NRST_STOP)
- Option byte load reset

5.1.1.1. Software Reset

This can be determined by looking at the reset flag in the RCC Clock Control and Status Register (RCC_CSR). To perform a software reset of the device, the SYSRESETREQ position 1 in the Cortex®-M4 Application Interrupt and Reset Control Register with FPU must be set.

5.1.1.2. Low-power management reset

There are two ways to trigger a low power management reset:

- generates a reset when it enters standby mode:

This reset is enabled by clearing the nRST_STDBY bit in the user option byte. When enabled, when the SLEEPDEEP signal is detected and the PWR_CR_PDDS register is configured for standby mode, the device will reset instead of entering standby mode.

- A reset is generated when entering stop mode:

This reset is enabled by clearing the nRST_STOP bit in the user option byte. When enabled, the device will reset instead of entering stop mode when the SLEEPDEEP signal is detected and the PWR_CR_PDDS register is configured for stop mode.

5.1.2. power reset

A power reset occurs whenever one of the following events occurs:

- Power-on/power-off reset (POR/PDR reset)
- On exit from standby mode

A power reset sets all registers to their reset values except the backup field.

These sources all act on the NRST pin, which remains low during the reset process.The RESET reset entry vector is fixed in the memory map at address 0x0000 0004.

The internal reset signal outputs a low-level pulse on the NRST pin. A pulse generator ensures that the reset pulse from each internal reset source lasts at least 20 μs. For external resets, the reset pulse is generated when the NRST pin is low.

5.1.3. Backup Domain Reset

A backup domain reset resets all backup domain registers to their respective reset values. The backup domain registers include the RCC_BDCR,backup registers, and the RTC section registers.

A backup domain reset occurs whenever one of the following events occurs:

- Software Reset: triggered by placing BDRST position 1 in the RCC Backup Domain Control Register (RCC_BDCR).
- After both the power supply VCC and VBAT have been powered down, either one of them is powered up again.

5.1.4. Unified handling of resets other than backup domain resets

Source resets other than backup domain resets act on the NRST pin, which is clocked low during the reset.

The circuit is shown below:

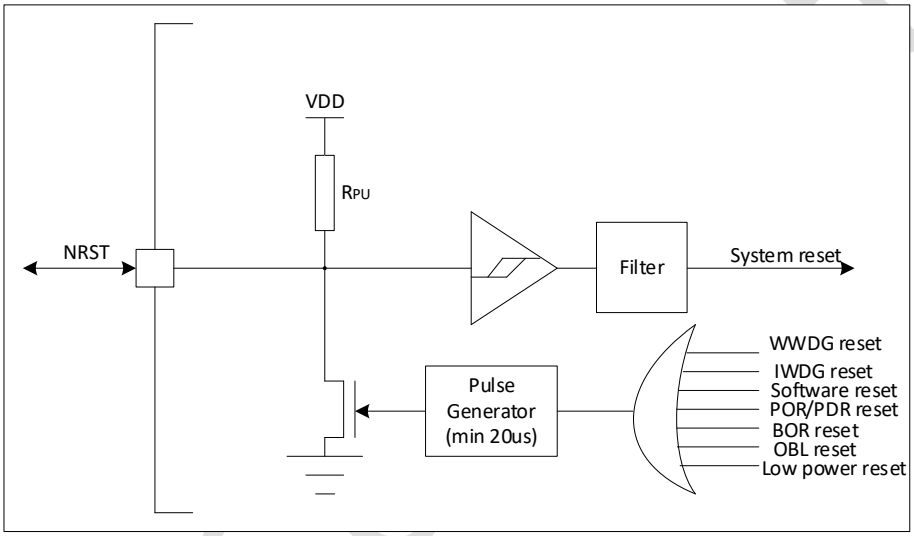


Figure 5-1 Reset circuit diagram

When an internal reset occurs (active high), the pulse generator starts generating a pulse signal that is guaranteed to last for at least 20us.This pulse signal energises the N-tube. The N-tube energisation continuously pulls down the NRST pin voltage, and when it is pulled down to VIL, a low level signal is generated at the NRST pin. This signal goes through the Schmitt trigger and filtering to produce a system reset (active low).

Note 1: Reset sources other than the external PIN reset source, i.e., the source of the pulse generator, will not be filtered for 30us before generating a system reset;

Note 2: The reset source of the pulse generator, after generating a system reset, will set the external pin reset flag in addition to setting the reset flag itself;

5.2. Functional description of the clock

5.2.1. Clock structure

The clock structure is shown below:

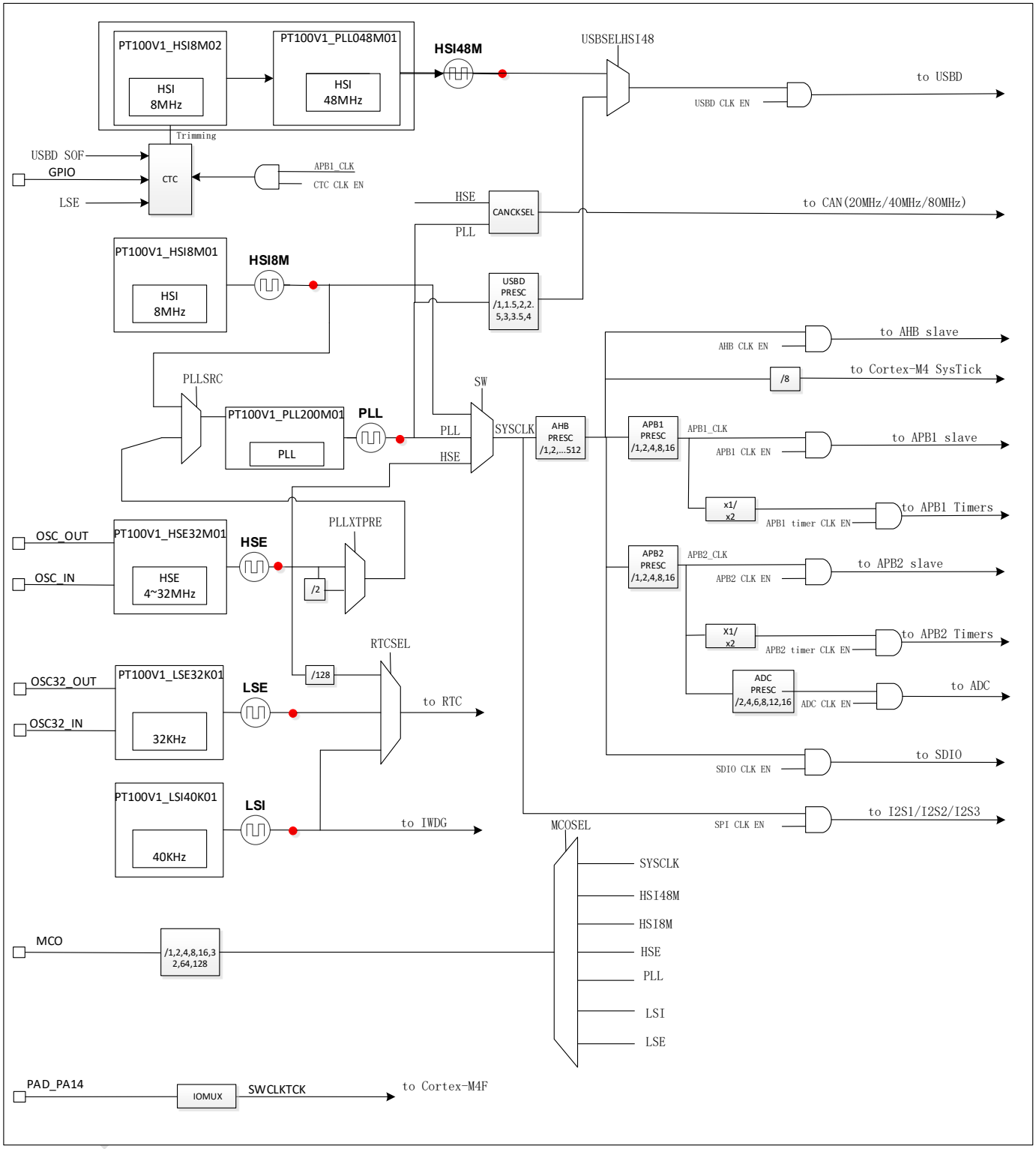


Figure 5-2 Clock structure

5.2.2. Clock source (clock signal and parameters subject to analogue module requirements)

The following clock sources exist in the system:

5.2.2.1. HSE Clock

External high frequency OSC. frequency range 4~32MHz.

The stabilisation time of the HSE clock is counted by the analogue HSE module and the stabilisation time is configured by FLASH_TRIM9.HSE_RDYSEL. The analogue HSE module generates the signal HSE_RDY after the HSE has started and stabilised.

The HSE clock can have two sources:

- External XTAL OSC + internal starting circuitry
- External clock input via OSC_IN IO (HSEBYP=1)

To ensure that the first clock out of the CLOCK is a stable clock, the HSE contains an internal counter that outputs the first rising edge of the clock after the count is complete. The counting period can be set by HSE_RDYSEL to select four counting gears, and the four counting gears select the appropriate counting period according to the HSE design. The counting period is selected according to the design of HSE. 00~10 counting positions are selected according to the design of HSE, 11 counting positions are test mode, no counting, and CLOCK outputs directly. normal mode and bypass mode have clock counting, but the counting period is different, and the counting period of the three counting positions in bypass mode is half of that in normal mode.

Bypass

HSE_BYP=1:

Bypass case, the analogue HSE is based on HSE_EXT_CLK and outputs the CLK_HSE i.e. HSE_RDY signal after counting half the time of the number of cycles configured by FLASH_TRIM9.HSE_RDYSEL.

5.2.2.2. HSI8M Clock

Internal 8MHz RC oscillator. Compared to the XTAL OSC, the RC OSC has low power consumption and short stabilisation time, but low accuracy. The HSI analogue module counts the stabilisation time, i.e. the HSI8M clock output to digital is the stabilisation clock.

After power-on reset, during the load phase, the calibration value of the HSI is loaded into the RCC_CR. HSICAL register.

Upon wake-up from stop mode, the HSI will be used as the system clock source.

5.2.2.3. HSI48M clock

The HSI48M clock consists of HSI_USB (8MHz) and PLL48M.The HSI48MHz clock supports self-calibration through the CTC module. The CTC calibrates the HSI_USB.The PLL48M reference clock is the output of the HSI_USB and is fixed to a frequency of 6x.The HSI48M is used as the clock for the USBD module.

5.2.2.4. PLL Clock

The PLL module is a frequency-doubled HSE clock with a PLL input clock frequency range of 8 MHz to 25 MHz and an output frequency of 24 MHz to 168 MHz.

5.2.2.5. LSE Clock

External 32.76KHz OSC used as a low power clock.

A balance between stabilisation time and power consumption can be made by configuring LSEDRV.The LSE stabilisation time is 0.5s (typical).The LSE stabilisation signal is generated by the LSE analogue module.

Similar to the HSE source, the LSE has two sources:

- 32.768K XTAL+ internal starting circuitry
- External clock input via OSC_IN (LSEBYP=1)

To ensure that the first clock out of the CLOCK is a stable clock, the LSE contains an internal counter that outputs the first rising edge of the clock after the count is complete. The counting period can be set by LSE_RDYSEL_VBKP to select four counting gears, in which the counting gears from 00 to 10 select the appropriate counting period according to the LSE design, and the counting gear from 11 is the test mode, no counting, and the CLOCK outputs it directly; the Normal mode and the bypass mode have the clock counting but the counting period is different, and the counting period of the three counting gears is the same as the counting period of the bypass mode, which is the same as the counting period of the normal mode. Normal mode and bypass mode both have clock counting, but the counting period is different, the counting period of bypass mode is half of that of normal mode.

Bypass

During LSE bypass, the HSERDY signal given by the analogue LSE has a value of zero.

In LSE bypass, the hardware VBAK_RCC module generates an LSE stabilisation wait time by counting, the length of which is controlled by the RCC_BDCR.LSERDYSEL_BP register.

In the non-bypass case, by configuring the FLASH information area register, it is possible to control the analogue output to the logic to clock CLK_LSE as a non-stable clock or as a stable clock, i.e., the LSE clock output through the MCO contains can contain either the non-stable phase or only the post-stable clock. Its ready signal is also given synchronously at the clock output.

In the bypass case, the clock output from the MCO is the original external clock and is not controlled by the LSE_ON signal. However, logic using the LSE clock will be controlled by the LSE_ON signal.

5.2.2.6. LSI Clock

Internal low frequency 40KHz clock. Used as the IWDG clock.

5.3. Register (base : 0x40021000)

5.3.1. Clock Control Register (RCC_CR)

Address:0x00

Reset value:0x0080 2083

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						PLL RDY	PLL ON	HSICAL[10:8]			Res	CSS ON	HSE BYP	HSE RDY	HSE ON
						R	RW	R				RS	RW	R	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HSICAL[7:0]								HSITRIM[4:0]					Res	HSI RDY	HSI ON
R								RW						R	RW

Bit	Name	R/W	Reset Value	Function
31:26	Reserved	-	-	Reserved
25	PLLRDY	R	0	PLL clock ready flag. Hardware set to indicate that the PLL clock is LOCKED. 0: PLL unlocked; 1: PLL locked;
24	PLLON	RW	0	PLL Enable. The hardware clears this bit when the system enters stop, standby mode. This bit cannot be cleared when the PLL clock is being used (SWS is the PLL) or will be used as the system clock (SW selects the PLL). 0: PLL OFF; 1: PLL ON;
23:21	HSICAL[10:8]	R	0x4	The HSI8M clock calibration value is 2 bits high. This register is loaded with the Flash information area value during the load phase of the power-up process. Software reads this register to return a value of HSICAL+HSITRIM, and when the HSITRIM value is changed, this register value is also updated.
20	Reserved	-	-	Reserved
19	CSSON	RW	0x0	HSE Clock Safety System Enable. When this bit is 1, the hardware enables the clock detection module if the HSE OSC is ready, or turns off the clock detection module if the HSE detection fails. 0: Clock safety system off (clock detection off); 1: clock safety system on (clock detection on if HSE clock is stable, otherwise clock detection off);
18	HSEBYP	RW	0	HSE shields the crystal and selects the pin input clock. This bit can only be written when HSEON=0. 0: HSE crystal is not shielded, external high-speed clock selects the crystal; 1: HSE crystal shielded, external high-speed clock selects external pin input clock source;
17	HSERDY	R	0	HSE crystal clock ready flag. This bit is set by hardware to indicate that the HSE crystal is stable. 0: HSE crystal is not ready; 1: HSE crystal is ready;
16	HSEON	RW	0	HSE Crystal Enable. When the system enters the stop, standby mode, the hardware clears this bit to turn off the HSE crystal. This bit cannot be set to 0 when the HSE is used directly or indirectly as the system clock source. 0: HSE crystal OFF; 1: HSE crystal ON;
15:8	HSICAL[7:0]	R	0x20	HSI8M clock calibration value.

Bit	Name	R/W	Reset Value	Function
				This register is loaded with the Flash information area value during the load phase of the power-up process. Software reads this register to return a value of HSICAL+HSITRIM, and when the HSITRIM value is changed, this register value is also updated.
7:3	HSITRIM[4:0]	RW	0x10	HSI8M clock Trimming value. Software writes to this register to adjust the HSI8M clock, which is superimposed on HSICAL and output to the analogue HSI8M. A system reset resets this register.
2	Reserved	-	-	Reserved
1	HSIRDY	R	1	HSI8M clock ready flag. Hardware setting indicates that the HSI OSC is stable. This bit is only valid when HSION=1. 0: HSI8M OSC not ready; 1: HSI8M OSC ready; When HSION is cleared, HSIRDY will be pulled low after 6 HSI8M clocks.
0	HSION	RW	1	HSI8M Clock Enable. The hardware will clear this register as necessary to stop the HSI8M when it enters stop, standby mode. This register cannot be 0 when the HSI8M is used directly or indirectly as the system clock. 0: HSI8M OSC OFF; 1: HSI8M OSC ON; The hardware will enable HSI8M in the following cases: 1) After the hardware wakes up from stop or standby mode; 2) HSE is directly/indirectly used as the system clock but there is a HSE CSS fail;

5.3.2. Clock Configuration Register (RCC_CFGR)

Address:0x04

Reset value:0x0000 00000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
USBPRE[2]	PLLMULL[5:4]		ADCPRE[2]	Res	MCO [2:0]			USBPRE[1:0]		PLLMULL[3:0]				PLLXTPRE	PLLSRC
RW	RW		RW		RW			RW		RW				RW	RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADCPRE[1:0]	PPRE2[2:0]				PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
RW	RW				RW			RW				R		RW	

Bit	Name	R/W	Reset Value	Function
31	USBPRE[2]	RW	0	Combine with bit 23:22 to configure the USB clock pre-divider value.

Bit	Name	R/W	Reset Value	Function
30:29	PLLMULL[5:4]	RW	0	Combine with bit 21:18 to configure the PLL multiplication factor.
28	ADCPRE[2]	RW	0	Combine with bit15:14 to configure the ADC clock division factor.
27	Reserved			Reserved
26:24	MCO [2:0]	RW	0	MCO output clock selection. 000: no clock, MCO output disabled 001: LSE32K 010: LSI40K 011: HSI48M 100: SYSCLK 101: HSI8M01 110: HSE 111: PLL others: reserved Note 1: If HSE and LSE are configured with HSE_RDYSEL and LSE_RDYSEL registers, the clock during startup can be output; Note 2: When selecting the system clock for the MCO output, it is necessary to ensure that the frequency at the output does not exceed the maximum allowable frequency of the IO.
23:22	USBPRE[1:0]	RW	0	USB clock pre-scaler. The combination with bit31 is configured as follows: 000: PLL clock 1.5 division frequency 001: PLL clock 010: PLL clock 2.5 divisions 011: PLL clock 2 divisions 100: PLL clock 3 divisions 101: PLL clock 3.5 divisions 11x: PLL clock 4 divisions
21:18	PLLMULL[3:0]	RW	0	Configure the PLL multiplication factor in combination with bit31:29. 000000：x2 000001：x3 000010：x4 ….. 010001：x19 010010：x20 010011：x21 Others: Reserved
17	PLLXTPRE	RW	0	The HSE divider is used as the PLL source.

Bit	Name	R/W	Reset Value	Function
				This bit can only be written when the PLL is turned off. 0: HSE clock as PLL source 1: HSE clock 2-division as PLL source
16	PLLSRC	RW	0	PLL clock source selection. This bit can only be written when the PLL is turned off. 0: res 1: PLLXTPRE configured HSE clock as PLL input clock
15:14	ADCPRE[1:0]	RW	0	Combine with bit27 to configure the frequency division factor of the ADC clock. 000: APB2 CLK 2 division frequency 001: APB2 CLK 4-division frequency 010: APB2 CLK 6 division frequency 011: APB2 CLK 8 division frequency 100：APB2 CLK 2 crossover frequency 101：APB2 CLK 12 crossover frequency 110：APB2 CLK 8 crossover frequency 111：APB2 CLK 16 crossover frequency
13:11	PPRE2[2:0]	RW	0	High-speed APB (APB2) clock division factor. PCLK is based on the HCLK division factor. 0xx: No frequency division 100: 2-division frequency 101: 4-division frequency 110: 8-division frequency 111: 16 crossover frequency
10:8	PPRE1[2:0]	RW	0	Low-speed APB (APB1) clock division factor. PCLK is based on the HCLK division factor. 0xx: No frequency division 100: 2-division frequency 101: 4-division frequency 110: 8-division frequency 111: 16 crossover frequency
7:4	HPRE[3:0]	RW	0	The AHB clock HCLK is based on the division factor of SYSCLK. 0xxx: no frequency division 1000: 2-division frequency 1001: 4-division frequency 1010: 8 divisions 1011: 16 divisions 1100: 64 crossover 1101: 128 crossover frequency

Bit	Name	R/W	Reset Value	Function
				1110: 256 crossover frequency 1111: 512dB In order to ensure the normal operation of the system, it is necessary to configure the appropriate frequency according to the VR power supply. Note: When the pre-divided frequency factor of the AHB clock is greater than 1, the prefetch buffer must be turned on. Note: It is recommended to switch the dividing factor step by step.
3:2	SWS[1:0]	R	0	System Clock Source Selection. This bit is controlled by hardware and indicates the selection of the system clock source. 00: HSI8M 01: HSE 10: PLL CLK Others: Reserved
1:0	SW[1:0]	RW	0	System Clock Source Selection. This bit is controlled by hardware and software and indicates the selection of the system clock source 00: HSI8M 01: HSE 10: PLL CLK Others: Reserved

5.3.3. Clock interrupt register (RCC_CIR)

Address:0x08

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								CSSC	Res	HSI48RDYC	PLLRDYC	HSERDYC	HSERDYC	LSERDYC	LSIRDYC
								W		W	W	W	W	W	W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		HSI48RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Res	HSI48RDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
		RW	RW	RW	RW	RW	RW	R		R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:24	Reserved	-	-	Reserved
23	CSSC	W	0	HSE Clock Safety System (CSS) interrupt flag cleared to zero. 0: No effect; 1: Clear the CSSF flag
22	Reserved	-	-	Reserved
21	HSI48RDYC	W	0	HSI48M ready interrupt flag cleared. 0: No effect; 1: Clear the HSI48RDYF flag;
20	PLLRDYC	W	0	PLL ready interrupt flag cleared. 0: No effect; 1: Clear PLLRDYF flag;
19	HSERDYC	W	0	HSE ready interrupt flag cleared. 0: No effect; 1: Clear the HSERDYF flag;
18	HSIRDYC	W	0	HSI ready interrupt flag cleared. 0: No effect; 1: Clear the HSIRDYF flag;
17	LSERDYC	W	0	LSE ready interrupt flag cleared. 0: No effect; 1: Clear the LSERDYF flag;
16	LSIRDYC	W	0	LSI ready interrupt flag cleared. 0: No effect; 1: Clear the LSIRDYF flag;
15:14	Reserved	-	-	Reserved
13	HSI48RDYIE	RW	0	HSI48M clockready interrupt enable. 0: disable; 1: enable;
12	PLLRDYIE	RW	0	PLL ready interrupt enable. 0: disable; 1: enable;
11	HSERDYIE	RW	0	HSE clock ready interrupt enable. 0: disable; 1: enable;
10	HSIRDYIE	RW	0	HSI8M clockready interrupt enable. 0: disable; 1: enable;
9	LSERDYIE	RW	0	LSE clock ready interrupt enable. 0: disable; 1: enable;
8	LSIRDYIE	RW	0	LSI clock ready interrupt enable. 0: disable; 1: enable;
7	CSSF	R	0	Clock Safety System Interrupt Flag.

Bit	Name	R/W	Reset Value	Function
				When the HSE clock fails, it is set by hardware to 1. Software clears this flag bit by writing CSSC bit 1. 0: No safety system interrupt generated by HSE clock failure is generated; 1: Safety system interrupt generated by HSE clock failure is generated;
6	Reserved		-	Reserved
5	HSI48RDYF	R	0	HSI48M clock ready interrupt flag. Hardware sets this register when the HSI48M clock is stable and HSI48RDYIE=1. 0: HSI48M clock ready interrupt not generated; 1: HSI48M clock ready interrupt generated; Write HSI48RDYC register 1 to clear this bit.
4	PLLRDYF	R	0	PLL ready interrupt flag. Hardware sets this register when PLL lock and PLLRDYDIE=1. 0: PLL lock interrupt not generated; 1: PLL lock interrupt generated; Write PLLRDYC1 to clear this bit.
3	HSERDYF	R	0	HSE clock ready interrupt flag. Hardware sets this register when the HSE clock is stable and HSERDYIE=1. 0: HSE clock ready interrupt not generated; 1: HSE clock ready interrupt generated; Write HSERDYC register 1 to clear this bit.
2	HSIRDYF	R	0	HSI8M clock ready interrupt flag. Hardware sets this register when the HSI8M clock is stable and HSIRDYIE=1. 0: HSI8M clock ready interrupt not generated; 1: HSI8M clock ready interrupt generated; Write HSIRDYC register 1 to clear this bit.
1	LSERDYF	R	0	LSERDY clock ready interrupt flag. Hardware sets this register when the LSE clock is stable and LSERDYDIE=1. 0: LSERDY clock ready interrupt not generated; 1: LSERDY clock ready interrupt generated; Write LSERDYC register 1 to clear this bit.
0	LSIRDYF	R	0	LSIRDY clock ready interrupt flag. Hardware sets this register when the LSI clock is stable and LSIRDYIE=1. 0: LSIRDY clock ready interrupt not generated; 1: LSIRDY clock ready interrupt generated; Write LSIRDYC register 1 to clear this bit.

5.3.4. APB2 Peripheral Reset Register (RCC_APB2RSTR)

Address:0x0C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										TIM11RST	TIM10RST	TIM9RST	Res		
										RW	RW	RW			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADC3RST	USART1RST	TIM8RST	SPI1RST	TIM1RST	ADC2RST	ADC1RST	Res								SYSCFGRST
RW	RW	RW	RW	RW	RW	RW									RW

Bit	Name	R/W	Reset Value	Function
31:22	Reserved		-	Reserved
21	TIM11RST	RW	0	TIM11 module reset. 0: no effect; 1: the module is reset;
20	TIM10RST	RW	0	TIM10 module reset. 0: no effect; 1: the module is reset;
19	TIM9RST	RW	0	TIM9 module reset. 0: no effect; 1: the module is reset;
18:16	Reserved	RES	-	Reserved
15	ADC3RST	RW	0	ADC3 module reset. 0: no effect; 1: the module is reset;
14	USART1RST	RW	0	USART1 module reset. 0: no effect; 1: the module is reset;
13	TIM8RST	RW	0	TIM8 module reset. 0: no effect; 1: the module is reset;
12	SPI1RST	RW	0	SPI1 module reset. 0: no effect; 1: the module is reset;
11	TIM1RST	RW	0	TIM1 module reset. 0: no effect; 1: the module is reset;
10	ADC2RST	RW	0	ADC2 module reset. 0: no effect; 1: the module is reset;
9	ADC1RST	RW	0	ADC1 module reset. 0: no effect; 1: the module is reset;
8:1	Reserved	-	-	Reserved
0	SYSCFGRST	RW	0	SYSCFG module reset.

Bit	Name	R/W	Reset Value	Function
				0: no effect; 1: the module is reset;

5.3.5. APB1 Peripheral Reset Register (RCC_APB1RSTR)

Address:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTCRST	Res	Res	PWRRST	Res		CANRST	Res	USBRST	I2C2RST	I2C1RST	USART5RST	USART4RST	USART3RST	USART2RST	Res
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**
SPI3RST	SPI2RST	Res		WWDGRST	Res		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
RW	RW			RW			RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	CTCRST	RW	0	CTC module reset. 0: no effect; 1: the module is reset;
30:29	Reserved	RES	-	Reserved
28	PWRRST	RW	0	Power interface module reset. 0: no effect; 1: the module is reset;
27:26	Reserved	RES	-	Reserved
25	CANRST	RW	0	CAN module reset. 0: no effect; 1: the module is reset;
24	Reserved	RES	-	Reserved
23	USBRST	RW	0	USBD module reset. 0: no effect; 1: the module is reset;
22	I2C2RST	RW	0	I2C2 module reset. 0: no effect; 1: the module is reset;
21	I2C1RST	RW	0	I2C1 module reset. 0: no effect; 1: the module is reset;
20	USART5RST	RW	0	UART5 module reset. 0: no effect; 1: the module is reset;
19	USART4RST	RW	0	USART4 module reset. 0: no effect;

Bit	Name	R/W	Reset Value	Function
				1: the module is reset;
18	USART3RST	RW	0	USART3 module reset. 0: no effect; 1: the module is reset;
17	USART2RST	RW	0	USART2 module reset. 0: no effect; 1: the module is reset;
16	Reserved	RES	-	Reserved
15	SPI3RST	RW	0	SPI3 module reset. 0: no effect; 1: the module is reset;
14	SPI2RST	RW	0	SPI2 module reset. 0: no effect; 1: the module is reset;
13:12	Reserved	RES	-	Reserved
11	WWDGRST	RW	0	WWDG module reset. 0: no effect; 1: the module is reset;
10:9	Reserved	RES	-	Reserved
8	TIM14RST	RW	0	TIM14 module reset. 0: no effect; 1: the module is reset;
7	TIM13RST	RW	0	TIM13 module reset. 0: no effect; 1: the module is reset;
6	TIM12RST	RW	0	TIM12 module reset. 0: no effect; 1: the module is reset;
5	TIM7RST	RW	0	TIM7 module reset. 0: no effect; 1: the module is reset;
4	TIM6RST	RW	0	TIM6 module reset. 0: no effect; 1: the module is reset;
3	TIM5RST	RW	0	TIM5 module reset. 0: no effect; 1: the module is reset;
2	TIM4RST	RW	0	TIM4 module reset. 0: no effect; 1: the module is reset;
1	TIM3RST	RW	0	TIM3 module reset. 0: no effect; 1: the module is reset;
0	TIM2RST	RW	0	TIM2 module reset. 0: no effect;

Bit	Name	R/W	Reset Value	Function
				1: the module is reset;

5.3.6. AHB1 Peripheral Clock Enable Register (RCC_AHB1ENR)

Address:0x14

Reset value:0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			ESMCEN	Res	SDIOEN	Res			CRCEN	Res	FMCEN	Res	SRAMEN	DMA2EN	DMA1EN
			RW		RW				RW		RW		RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:13	Reserved	RES	-	Reserved
12	ESMCEN	RW	0	ESMC(QSPI) module clock enable. 0: disable; 1: enable;
11	Reserved	RES	-	Reserved
10	SDIOEN	RW	0	SDIO module clock enable. 0: disable; 1: enable;
9:7	Reserved	RES	-	Reserved
6	CRCEN	RW	0	CRC module clock enable. 0: disable; 1: enable;
5	Reserved	RES	-	Reserved
4	FMCEN	RW	1	FLASH interface module clock enable for sleep mode. 0: disable; 1: enable;
3	Reserved	RES	-	Reserved
2	SRAMEN	RW	1	SRAM memory area clock enable for sleep mode. 0: disable; 1: enable;
1	DMA2EN	RW	0	DMA2 module clock enable. 0: disable; 1: enable;
0	DMA1EN	RW	0	DMA1 module clock enable. 0: disable; 1: enable;

5.3.7. APB2 Peripheral Clock Enable Register (RCC_APB2ENR)

Address:0x18

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										TIM11EN	TIM10EN	TIM9EN	Res		
										RW	RW	RW			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADC3EN	USART1EN	TIM8EN	SPI1EN	TIM1EN	ADC2EN	ADC1EN	Res								SYSCFGEN
RW	RW	RW	RW	RW	RW	RW									RW

Bit	Name	R/W	Reset Value	Function
31:22	Reserved	RES	-	Reserved
21	TIM11EN	RW	0	TIM11 module clock enable. 0: disable; 1: enable;
20	TIM10EN	RW	0	TIM10 module clock enable. 0: disable; 1: enable;
19	TIM9EN	RW	0	TIM9 module clock enable. 0: disable; 1: enable;
18:16	Reserved	RES	-	Reserved
15	ADC3EN	RW	0	ADC3 module clock enable. 0: disable; 1: enable;
14	USART1EN	RW	0	USART1 module clock enable. 0: disable; 1: enable;
13	TIM8EN	RW	0	TIM8 module clock enable. 0: disable; 1: enable;
12	SPI1EN	RW	0	SPI1 module clock enable. 0: disable; 1: enable;
11	TIM1EN	RW	0	TIM1 module clock enable. 0: disable; 1: enable;
10	ADC2EN	RW	0	ADC2 module clock enable. 0: disable; 1: enable;
9	ADC1EN	RW	0	ADC1 module clock enable. 0: disable; 1: enable;
8:1	Reserved	RES	-	Reserved

Bit	Name	R/W	Reset Value	Function
0	SYSCFGEN	RW	0	SYSCFG module clock enable. 0: disable; 1: enable;

5.3.8. APB1 Peripheral Clock Enable Register (RCC_APB1ENR)

Address:0x1C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTCEN	Res	Res	PWREN	BKPEN	Res	CANEN	Res	USBDEN	I2C2EN	I2C1EN	USART5EN	USART4EN	USART3EN	USART2EN	Res
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**
SPI3EN	SPI2EN	Res		WWDGEN	Res		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
RW	RW			RW			RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	CTCEN	RW	0	CTC module clock enable. 0: disable; 1: enable;
30:29	Reserved	RES	-	Reserved
28	PWREN	RW	0	PWR module clock enable. 0: disable; 1: enable;
27	BKPEN	RW	0	BACKUP module clock enable. 0: disable; 1: enable;
26	Reserved	RES	-	Reserved
25	CANEN	RW	0	CAN module clock enable. 0: disable; 1: enable;
24	Reserved	RES	-	Reserved
23	USBDEN	RW	0	USBD module clock enable. 0: disable; 1: enable;
22	I2C2EN	RW	0	I2C2 module clock enable. 0: disable; 1: enable;
21	I2C1EN	RW	0	I2C1 module clock enable. 0: disable; 1: enable;
20	USART5EN	RW	0	UART5 module clock enable. 0: disable; 1: enable;

Bit	Name	R/W	Reset Value	Function
19	USART4EN	RW	0	UART4 module clock enable. 0: disable; 1: enable;
18	USART3EN	RW	0	USART3 module clock enable. 0: disable; 1: enable;
17	USART2EN	RW	0	USART2 module clock enable. 0: disable; 1: enable;
16	Reserved		-	Reserved
15	SPI3EN	RW	0	SPI3 module clock enable. 0: disable; 1: enable;
14	SPI2EN	RW	0	SPI2 module clock enable. 0: disable; 1: enable;
13:12	Reserved	RES	-	Reserved
11	WWDGEN	RW	0	WWDG module clock enable. 0: disable; 1: enable;
10:9	Reserved		-	Reserved
8	TIM14EN	RW	0	TIM14 module clock enable. 0: disable; 1: enable;
7	TIM13EN	RW	0	TIM13 module clock enable. 0: disable; 1: enable;
6	TIM12EN	RW	0	TIM12 module clock enable. 0: disable; 1: enable;
5	TIM7EN	RW	0	TIM7 module clock enable. 0: disable; 1: enable;
4	TIM6EN	RW	0	TIM6 module clock enable. 0: disable; 1: enable;
3	TIM5EN	RW	0	TIM5 module clock enable. 0: disable; 1: enable;
2	TIM4EN	RW	0	TIM4 module clock enable. 0: disable; 1: enable;
1	TIM3EN	RW	0	TIM3 module clock enable. 0: disable; 1: enable;

Bit	Name	R/W	Reset Value	Function
0	TIM2EN	RW	0	TIM2 module clock enable. 0: disable; 1: enable;

5.3.9. RTC domain control register (RCC_BDCR)

Address:0x20

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															BDRST
															RW
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RTCEN	Res				RSTOUT_DIS	RTCSEL[1:0]		Res	LSERDYSEL_BP[1:0]		LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
RW					RW	RW			RW		RW		RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:17	Reserved		-	Reserved
16	BDRST	RW	0	Backup domain soft reset. 0: no effect; 1: reset;
15	RTCEN	RW	0	RTC and TAMP clock enable. 0: disable; 1: enable;
14:11	Reserved		-	Reserved
10	RSTOUT_DIS	RW	0	Reset output disable register. 0: Internal reset output after 30us pulse generator; 1: internal reset disable output. This bit needs to be configured to 1 when the standby mode is entered in order to ensure the wake-up time.
9:8	RTCSEL[1:0]	RW	0	RTC clock source selection. 00: No clock 01: LSE 10: LSI 11: HSE divided by 128. Once the RTC clock source is selected it cannot be changed unless the backup domain reset is cleared.
7	Reserved		-	Reserved
6:5	LSERDYSEL_BP[1:0]	RW	0x0	Count time selection on LSE Bypass. 00: 8 LSE clocks 01: 16 LSE clocks 10: 24 LSE clocks 11: 32 LSE clocks

Bit	Name	R/W	Reset Value	Function
4:3	LSEDRV[1:0]	RW	0	LSE Drive Capability Setting, default 00 00：gm 2.5uA/V 01：gm 3.75uA/V 10：gm 8.5uA/V 11：gm 13.5uA/V
2	LSEBYP	RW	0	LSE OSC bypass 0: Not bypassed，Low-speed external clock selection crystal； 1: Bypassed，Low-speed external clock selection external interface input clock； Note: This bit can only be written if the external 32KHz OSC is disabled (LSEON=0 and LSERDY=0).
1	LSERDY	R	0	LSE OSC ready. Hardware configuration of this bit to 1 indicates that the LSE clock is ready. After LSEON is cleared, this bit takes 6 LSE clocks before it is cleared.
0	LSEON	RW	0	LSE OSC enable. 0: disable; 1: enable;

5.3.10. Control/status register (RCC_CSR)

Address:0x24

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PWRRSTF	PINRSTF	OBLRSTF	RMVF	Res							
R	R	R	R	R	R	R	R								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														LSIRDY	LSION
														R	RW

Bit	Name	R/W	Reset Value	Function
31	LPWRRSTF	R	0	Low Power Reset Flag. Hardware sets this register when an illegal stop/standby low power mode is entered. This register can only be operated when the nRST_STOP, nRST_STDBY option bits are in the clear (valid) state. RMVF set to 1 clears this bit.

Bit	Name	R/W	Reset Value	Function
30	WWDGRSTF	R	0	Window WDG Reset Flag. RMVF set to 1 clears this bit.
29	IWDGRSTF	R	0	IWDG Reset Flag. RMVF set to 1 clears this bit.
28	SFTRSTF	R	0	Soft reset flag. RMVF set to 1 clears this bit.
27	PWRRSTF	R	1	POR/PDR Reset Flag. RMVF set to 1 clears this bit.
26	PINRSTF	R	0	External NRST pin reset flag. RMVF set to 1 clears this bit.
25	OBLRSTF	R	0	Option byte loader Reset Flag. RMVF set to 1 clears this bit.
24	RMVF	RW	0	Software set bit clears the reset flag. An operation that writes this bit to 1 clears the reset flag starting at bit 25, which will remain at 1 after a software write of 1 and will not be cleared by hardware.
23:2	Reserved		-	Reserved
1	LSIRDY	R	0	LSI OSC Stability marker. LSIRDY is generated by the analogue LSI.
0	LSION	RW	0	LSI OSC Enable. 0 forbidden: stop; 1: enable; Hardware enable for analogue LSI: 1) Hardware IWDG enable;

5.3.11. Clock Reset Configuration Register 1 (RCC_CFGR1)

Address:0x24

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
USBSELHSI48	Res		HSI48CAL[12:0]												
RW			RW												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSI48TRIM[6:0]							Res			HSI48RDY	HSI48ON	Res	MCOPRE[2:0]		
RW										R	RW		RW		

Bit	Name	R/W	Reset Value	Function
31	USBSELHSI48	RW	0	USBD module clock (not system clock) selection. 0: USBD clock selection PLL (divided frequency); 1: USBD clock selection HSI48M;
30:29	Reserved			
28:16	HSI48CAL[12:0]	R	0x1080	HSI48M clock calibration value.

Bit	Name	R/W	Reset Value	Function
				This register is loaded with the Flash information area value during the load phase of the power-up process. Software reads this register to return a value of HSI48CAL + HSI48TRIM, and when the HSI48TRIM value is changed, this register value is also updated.
15:9	HSI48TRIM[6:0]	RW	0x40	HSI48M Clock Trimming Value. Software writes to this register to adjust the HSI48M clock, which is superimposed on the HSI48CAL and output to the analogue HSI48M. This register is reset on system reset.
8:6	Reserved		-	Reserved
5	HSI48RDY	R	0	HSI48M clock ready flag. Hardware setting indicates that the HSI48M is stable. This bit is only valid when HSI48MON=1. 0: HSI48M not ready; 1: HSI48M ready; When HSI48MON is cleared, HSI48MRDY will be pulled low after 6 HSI48M clocks.
4	HSI48ON	RW	0	HSI48M Clock Enable. The hardware will clear this register to stop the HSI48M as needed when it enters stop, standby mode. 0: HSI48M OFF; 1: HSI48M ON;
3	Reserved		-	Reserved
2:0	MCOPRE[2:0]	RW	0	MCO (microcontroller clock output) crossover coefficients. Software controls these bits to set the crossover coefficient of the MCO output: 000：1 001：2 010：4 011：8 100：16 101：32 110：64 111：128 It is recommended that these bits be set before the MCO output is enabled.

5.3.12. AHB1 peripheral reset register (RCC_AHB1RSTR)

Address:0x2C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			ESMCRST	Res	SDIORST	Res			CRCRST	Res				DMA2RST	DMA1RST
			RW		RW				RW					RW	RW

Bit	Name	R/W	Reset Value	Function
31:13	Reserved		-	Reserved
12	ESMCRST	RW	0	ESMC(QSPI) module reset enable. 0: disable; 1: enable;
11	Reserved		-	Reserved
10	SDIORST	RW	0	SDIO module reset enable. 0: disable; 1: enable;
9:7	Reserved		-	Reserved
6	CRCRST	RW	0	CRC module reset enable. 0: disable; 1: enable;
5:2	Reserved	RES	-	Reserved
1	DMA2RST	RW	0	DMA2 module reset enable. 0: disable; 1: enable;
0	DMA1RST	RW	0	DMA1 module reset enable. 0: disable; 1: enable;

5.3.13. AHB2 peripheral reset register (RCC_AHB2RSTR)

Address:0x30

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Res	
									RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31:7	Reserved		-	Reserved
6	IOPERST	RW	0	Module reset. 0: no effect;

Bit	Name	R/W	Reset Value	Function
				1: the module is reset;
5	IOPDRST	RW	0	Module reset. 0: no effect; 1: the module is reset;
4	IOPCRST	RW	0	Module reset. 0: no effect; 1: the module is reset;
3	IOPBRST	RW	0	Module reset. 0: no effect; 1: the module is reset;
2	IOPARST	RW	0	Module reset. 0: no effect; 1: the module is reset;
1:0	Reserved		-	Reserved

5.3.14. AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address:0x34

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res									IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res	
									RW	RW	RW	RW	RW		

Bit	Name	R/W	Reset Value	Function
31:7	Reserved		-	Reserved
6	IOPEEN	RW	0	IOPE module clock enable. 0: disable; 1: enable;
5	IOPDEN	RW	0	IOPD module clock enable. 0: disable; 1: enable;
4	IOPCEN	RW	0	IOPC module clock enable. 0: disable; 1: enable;
3	IOPBEN	RW	0	IOPB module clock enable. 0: disable; 1: enable;
2	IOPAEN	RW	0	IOPA module clock enable. 0: disable; 1: enable;
1:0	Reserved	RES	-	Reserved

5.3.15. Clock Reset Configuration Register 2 (RCC_CFGR2)

Address:0x38

Reset value:0x0000 0008

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

Bit	Name	R/W	Reset Value	Function
31:10	Reserved		-	Reserved
9:8	HSE_DRV[1:0]	RW	0x0	HSE Drive Capability Configuration 00: gm 3.5mA/V 01: gm 5.0mA/V 10: gm 7.5mA/V 11: gm 10mA/V
7:4	Reserved		-	Reserved
3:0	CANCKSEL[3:0]	RW	0x8	CAN communication clock selection. 0000: PLL clock; 0001: PLL clock 2 divisions; 0010: PLL clock 3 division frequency; 0011: PLL clock 4 division frequency; 0100: PLL clock 5 division frequency; 0101: PLL clock 6 division frequency; 0110: PLL clock 7 division frequency; 0111: PLL clock 8 division frequency; 1000: HSE clock; others: no clock;

5.3.16. RCC register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40021000	RCC_CR	Reserved						PLLRDY	PLLON	HSICAL[10:8]			Reserved	CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]								HSITRIM[4:0]					Reserved	HSIRDY	HSION
	Reset Value							0	0	1	0	0		0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0		1	1
0x04	RCC_CFGR	USBPRE[2]	PLLMULL[5:4]	ADCPRE[2]	MCO[3:0]			USBPRE[1:0]		PLLMULL[3:0]						PLLXTPRE	PLLSRC	ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	RCC_CIR	Reserved								CSSC	Reserved	HSI48RDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved		HSI48RDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved	HSI48RDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset Value									0		0	0	0	0	0	0			0	0	0	0	0	0	0		0	0	0	0	0	0
0x0C	RCC_APB2RSTR	Reserved										TIM11RST	TIM10RST	TIM9RST	Reserved			ADC3RST	USART1RST	TIM8RST	SPI1RST	TIM1RST	ADC2RST	ADC1RST	Reserved								SYSCFGRST
	Reset Value											0	0	0				0	0	0	0	0	0	0									0
0x10	RCC_APB1RSTR	CTCRST	Reserved	Reserved	PWRRST	BKPRST	Reserved	CANRST	Reserved	USBRST	I2C2RST	I2C1RST	USART5RST	USART4RST	USART3RST	USART2RST	Reserved	SPI3RST	SPI2RST	Reserved		WWDGTRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset Value	0			0	0		0		0	0	0	0	0	0	0		0	0			0			0	0	0	0	0	0	0	0	0
0x14	RCC_AHB1ENR	Reserved																			ESMCEN	Reserved	SDIOEN	Reserved			CRCEN	Reserved	FMCEN	Reserved	SRAMEN	DMA2EN	DMA1EN
	Reset Value																				0		0				0		1		1	0	0
0x18	RCC_APB2ENR	Reserved										TIM11EN	TIM10EN	TIM9EN	Reserved			ADC3EN	USART1EN	TIM8EN	SPI1EN	TIM1EN	ADC2EN	ADC1EN	Reserved								SYSCFGEN
	Reset Value											0	0	0				0	0	0	0	0	0	0									0
0x1C	RCC_APB1ENR	CTCEN	Reserved	Reserved	PWREN	BKPEN	Reserved	CANEN	Reserved	USBEN	I2C2EN	I2C1EN	USART5EN	USART4EN	USART3RN	USART2EN	Reserved	SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset Value	0			0	0		0		0	0	0	0	0	0	0		0	0		0	0		0	0	0	0	0	1	0	1	0	0
0x20	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved				RTCSEL[1:0]		Reserved			LSEDRV[1:0]			LSEBYP	LSERDY	LSEON
	Reset Value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	OBLRSTF	RMVF	Reserved																					LSIRDY	LSION	
	Reset Value	0	0	0	0	1	1	0	0																						0	0	
0x28	RCC_CFGR1	USBSELHSI48	PVDSEL[1:0]		HSI48CAL[12:0]													HSI48TRIM[6:0]							Reserved			HSI48RDY	HSI48ON	Reserved		MCOPRE[2:0]	
	Reset Value	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0				0	0		0	0	0
0x2C	RCC_AHB1RSTR	Reserved																			ESMCRST	Reserved	SDIORST	Reserved			CRCRST	Reserved				DMA2RST	DMA1RST
	Reset Value																				0		0				0					0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	RCC_AHB2RSTR	Reserved																									IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Reserved	
	Reset Value																										0	0	0	0	0		
0x34	RCC_AHB2ENR	Reserved																									IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Reserved	
	Reset Value																										0	0	0	0	0		
0x38	RCC_CFGR2	Reserved																						HSE_DRV[1:0]		Reserved				CANCKSEL[3:0]			
	Reset Value																							0	0					1	0	0	0

6. Backup register (BKP)

6.1. Introduction

The backup registers are 42 16-bit registers that can be used to store 84 bytes of user application data. The module is in the backup domain and when the VDD power is cut off, they are still maintained powered by VBAT. They are also not reset when the system is woken up in standby mode, or during a system reset or power on reset (POR).

The backup registers are reset by a backup domain power-on reset (BPOR+BDRST) and the backup register contents are cleared when an intrusion event is detected.

In addition, the BKP control register is used to manage the intrusion detection and RTC calibration functions.

After reset, access to the backup registers and RTC is disabled and the backup domain is protected against possible accidental write operations. Access to the backup registers and RTC can be enabled by performing the following operations:

- Turn on the clock for the power and backup interfaces by setting the PWREN and BKPEN bits of register RCC_APB1ENR
- DBP bit of the Power Control Register (PWR_CR) to enable access to the backup registers and RTC.

6.1.1. Main features

- Supports 84-byte data back-up registers
- Status/control register used to manage anti-intrusion detection with interrupt capability
- Parity register used to store the RTC parity value.
- Outputs RTC calibration clock, RTC alarm pulse, or seconds pulse on pin PC13 (when this pin is not used for intrusion detection).

6.2. Functional description

6.2.1. Intrusion Detection

An Intrusion Detection event is generated when the signal on the TAMPER pin changes from 0 to 1 or from 1 to 0 (depending on the TPAL bit of the Backup Control Register BKP_CR). The intrusion detection event clears all data backup register contents. However, to avoid losing the intrusion event, the intrusion detection signal is a logical 'and' of the edge detected signal and the intrusion detection allow bit, so that intrusion events that occur before the intrusion detection pin is allowed can also be detected.

- When TPAL=0: If the TAMPER pin is already high before the Trespass Detect TAMPER pin is activated (by setting the TPE bit), an additional trespass event is generated as soon as the trespass detection function is activated (even though no rising edge occurs after the TPE position '1').

- When TPAL=1: If this pin is already low before activating the intrusion detection pin TAMPER (by setting the TPE bit), an additional intrusion event is generated once the intrusion detection function is activated (although no falling edge occurs after the TPE position '1').

Setting the TPIE bit of the BKP_CSR register to '1' generates an interrupt when an intrusion event is detected. After an intrusion event has been detected and cleared, the intrusion detection pin TAMPER should be disabled. The intrusion detection function is then re-enabled with the TPE bit before writing to the backup data register again. This prevents software from writing to the backup data register while there is still an intrusion event on the intrusion detection pin. This is equivalent to level detecting the intrusion pin TAMPER.

Note: The intrusion detection function remains active when the VCC power supply is disconnected. To avoid unnecessary resetting of the data backup register, the TAMPER pin should be connected off-chip to the correct level.

6.2.2. RTC Calibration

To facilitate measurements, the RTC clock can be output via a 64-division frequency to the intrusion detection pin TAMPER. This feature is enabled by setting the CCO bit of the RTC calibration register (BKP_RTCCR).

This clock can be slowed down by up to 121 ppm by configuring the CAL [6:0] bits.

6.2.2.1. Overview

Real-time clock (RTC) accuracy is a requirement for most embedded applications, but due to the external environment-temperature variations and changes in the clock's crystal frequency-RTC accuracy may not be as accurate as expected.

The RTC module contains digital clock calibration circuitry suitable for manufacturing environments, allowing applications to compensate for crystal and temperature variations.

6.2.2.2. RTC Calibration Methods

The RTC clock can be driven by an optional quartz crystal controlled oscillator rated at 32.768 kHz. The crystal oscillator is one of the most accurate circuits for providing a fixed frequency. Clock errors can occur for two reasons:

- Temperature changes
- Crystal changes

As mentioned earlier, most clock chips compensate for crystal frequency and temperature variations by using cumbersome trimming capacitors. The design uses periodic counter calibration. The digital calibration circuit eliminates 0 to 127 cycles every 220 clock cycles. The number of pulses that are cancelled depends on the value of the RTC clock calibration register BKP_RTCCR.CAL loaded into the BKP. Since the RTC clock calibration register is in the backup domain, the calibration value is not lost even if the device is powered down if the battery is connected to the VBAT pin.

The block diagram of the structure is shown below:

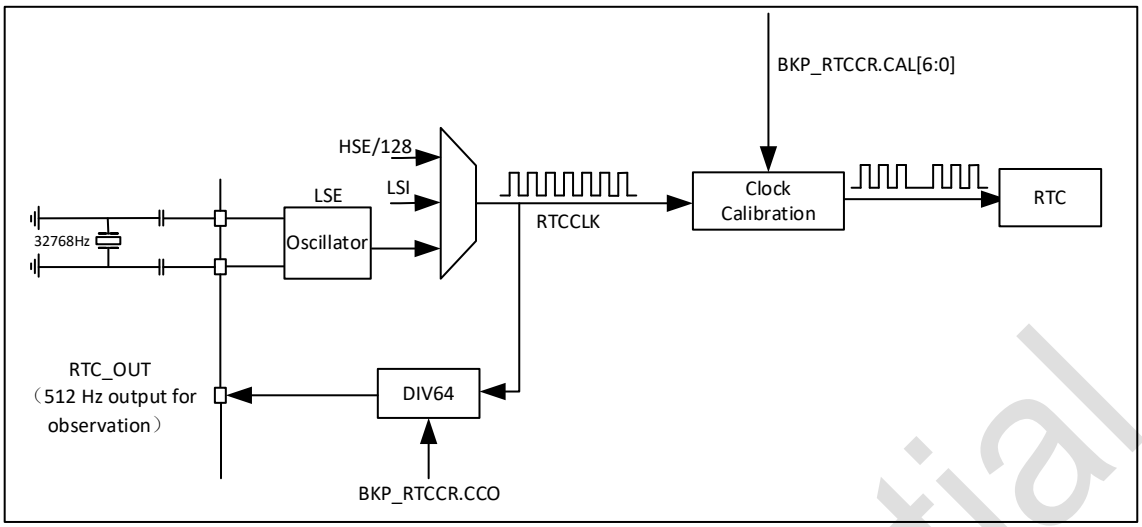


Figure 6-1 Block diagram of RTC structure

Note: The clock output on the pin is the RTC clock before calibration, so calibration does not change its value.

The effect of each calibration step is 1 oscillator cycle subtracted from every 1,048,576 (220) actual oscillator cycles. That is, the adjustment for each calibration step in the calibration register is 0.954 (1,000,000/220) ppm. therefore, the oscillator clock can be slowed from 0 to 121 ppm.

The following table shows the ppm and seconds that each bit represents in real time for each month (30 days).

Table 6-1 Real-time ppm and seconds

calibration value	Rounded to the nearest ppm	Values are in seconds, rounded to the nearest second for each month (30 days)	calibration value	Rounded to the nearest ppm	Values are in seconds, rounded to the nearest second for each month (30 days)
0	0	0	64	61	158
1	1	2	65	62	161
2	2	5	66	63	163
3	3	7	67	64	166
4	4	10	68	65	168
5	5	12	69	66	171
6	6	15	70	67	173
7	7	17	71	68	176
8	8	20	72	69	178
9	9	22	73	70	180
10	10	25	74	71	183
11	10	27	75	72	185
12	11	30	76	72	188
13	12	32	77	73	190
14	13	35	78	74	193

calibration value	Rounded to the nearest ppm	Values are in seconds, rounded to the nearest second for each month (30 days)	calibration value	Rounded to the nearest ppm	Values are in seconds, rounded to the nearest second for each month (30 days)
15	14	37	79	75	195
16	15	40	80	76	198
17	16	42	81	77	200
18	17	44	82	78	203
19	18	47	83	79	205
20	19	49	84	80	208
21	20	52	85	81	210
22	21	54	86	82	213
23	22	57	87	83	215
24	23	59	88	84	218
25	24	62	89	85	220
26	25	64	90	86	222
27	26	67	91	87	225
28	27	69	92	88	227
29	28	72	93	89	230
30	29	74	94	90	232
31	30	77	95	91	235
32	31	79	96	92	237
33	31	82	97	93	240
34	32	84	98	93	242
35	33	87	99	94	245
36	34	89	100	95	247
37	35	91	101	96	250
38	36	94	102	97	252
39	37	96	103	98	255
40	38	99	104	99	257
41	39	101	105	100	260
42	40	104	106	101	262
43	41	106	107	102	264
44	42	109	108	103	267
45	43	111	109	104	269
46	44	114	110	105	272
47	45	116	111	106	274
48	46	119	112	107	277
49	47	121	113	108	279
50	48	124	114	109	282
51	49	126	115	110	284
52	50	129	116	111	287
53	51	131	117	112	289

calibration value	Rounded to the nearest ppm	Values are in seconds, rounded to the nearest second for each month (30 days)	calibration value	Rounded to the nearest ppm	Values are in seconds, rounded to the nearest second for each month (30 days)
54	51	133	118	113	292
55	52	136	119	113	294
56	53	138	120	114	297
57	54	141	121	115	299
58	55	143	122	116	302
59	56	146	123	117	304
60	57	148	124	118	307
61	58	151	125	119	309
62	59	153	126	120	311
63	60	156	127	121	314

As mentioned above, the RTC clock calibration circuit only subtracts cycles from the crystal clock. Based on the default setting of the RTC prescaler value to 32768, faster crystal frequencies (>32768 Hz) can be calibrated, while slower crystal frequencies (<32768 Hz) cannot be compensated for (only faster crystals are calibrated, slower ones exacerbate the slowness). So only the crystal frequency range [32772,32768] can be calibrated.

Since the crystal frequency may vary by about 32.768 kHz, a solution can be considered that consists in setting the RTC prescaler to 32766 (instead of 32768). Thus, the crystal frequency is changed from 32768 to 32766. in this way, the crystal frequency in the range [32770,32766] can be compensated for.

For the remainder of this document, the RTC prescaler value considered will be 32766.

6.2.2.3. Calculate the amount of required calibration

In order to determine how much calibration is required in a given application, a methodology particularly suited to the manufacturing environment needs to be retained. It involves the use of the RTC clock output mode, which obtains a 512 Hz signal from the clock division chain. This signal can be used to measure the accuracy of the crystal oscillator.

The method can be divided into the following steps.

- Enable the Low Speed External Oscillator (LSE), select the LSE as the RTC clock source, and then enable the RTC clock.
- Enable the frequency of the RTC clock output to be divided by 64 on the RTC_OUT pin for crystal frequency measurement. This is done by setting the CCO bit in BKP_RTCCR.
- Calculate the crystal frequency deviation (ppm). The deviation in ppm can be quickly calculated by dividing the measured deviation of 511.968 Hz by 511.968 and multiplying the result by 1 million. Use Table 1 to find the nearest calibration value. This table is a straightforward lookup table for calibration values based on the change in value expressed in ppm.
- Load a calibration value in the RTC calibration register to compensate for crystal deviation.

Note: To set the RTC prescaler to 32766, write 32765 to the RTC prescaler load register.

For example, if the frequency measured in test mode is 511.982 Hz, then δ is 0.014. Dividing by 511.968 and multiplying by 1,000,000 results in 27.35 ppm. In this case, the closest compensated value is 28. The error will be reduced from 27.35 ppm (~71 seconds per month) to 0.65 ppm (~1.7 seconds per month).

Note: Since RTC calibration is based on removing clock cycles, it does not improve counts over short periods of time, it only improves counts over long periods of time. For example, counting 1/100th of a second with RTC without calibration will be more accurate than with calibration. Since the calibration cycle may or may not occur within the time frame under consideration, the resultant values may change significantly. Therefore, depending on the application, it is best to avoid using calibration.

6.3. BKP register (base address = 0x40006C00)

Note: The BKP register can be accessed as a half-word (16-bit) or word (32-bit).

6.3.1. Backup Data Register (BKP_DRx) (x=1…42)

Address offset：0x04~0x28,0x40~0xBC

Reset value：0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	-	Reserved
15:0	D[15:0]	RW	0	Backup data. These bits can be used to write user data. Note: The BKP_DRx register is not reset by a system reset, power reset, or wake from standby mode reset. It can be reset by a backup domain reset or (if the intrusion detection pin TAMPER function is enabled) by an intrusion pin event.

6.3.2. RTC Clock Calibration Register (BKP_RTCCR)

Address offset：0x2C

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						ASOS	ASOE	CCO	CAL[6:0]						
						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:10	Reserved	RW	-	Reserved
9	ASOS	RW	0	Alarm or Second Output Selection. When the ASOE bit is set, the ASOS bit can be used to select whether an RTC seconds pulse or an alarm clock pulse signal is output on the TAMPER pin. 0: Output RTC alarm clock pulse; 1: outputs a seconds pulse; Note: This bit can only be cleared by a backup area reset.
8	ASOE	RW	0	Allow the output of an alarm clock or seconds pulse. Depending on the ASOS bit configuration, this bit allows an RTC alarm clock or seconds pulse to be output to the TAMPER pin. The width of the output pulse is one RTC clock cycle. TAMPER cannot be enabled when the ASOE bit is set. Note: This bit can only be cleared by a backup area reset.
7	CCO	RW	0	Calibrate clock output. 0: No effect; 1: RTC 64-division clock output at the intrusion detection pin. After CCO position 1, the intrusion detection function must be turned off to avoid detecting useless intrusion signals. Note: This bit is cleared when the VCC supply is disconnected.
6:0	CAL[6:0]	RW	0	Calibration Value. The calibration value indicates how many clock pulses will be skipped within every 220 clock pulses. This can be used to calibrate the RTC to slow down the clock at a ratio of 1000000/220ppm. The RTC clock can be slowed down from 0 to 121 ppm.

Note: When CCO and ASOE are configured at the same time, ASOS has high priority.

6.3.3. Backup control register (BKP_CR)

Address offset：0x30

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPAL	TPE
														RW	RW

Bit	Name	R/W	Reset Value	Function
31:2	Reserved	RW	-	Reserved
1	TPAL	RW	0	Intrusion Detection TAMPER pin active level. 0: A high level on the Intrusion Detect TAMPER pin clears all data backup registers (with TPE=1); 1: A low level on the Intrusion Detect TAMPER pin clears all data backup registers (when TPE=1); Note: It is recommended to change the value of TPAL when TPE is 0, otherwise a false intrusion event will be generated.
0	TPE	RW	0	Activate the Intrusion Detection TAMPER pin. 0: Intrusion detection TAMPER pin is used as a general purpose IO port; 1: Intrusion detection TAMPER pin is used as intrusion detection;

6.3.4. Backup control/status register (BKP_CSR)

Address offset：0x34

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	TIF	TEF	Res.	Res.	Res.	Res.	Res.	TPIE	CTI	CTE
						R	R						RW	W	W

Bit	Name	R/W	Reset Value	Function
31:10	Reserved	RW	-	Reserved
9	TIF	R	0	Intrusion Interrupt Flag.

Bit	Name	R/W	Reset Value	Function
				This bit is set by hardware when an intrusion event is detected and the TPIE bit is 1. This flag bit is cleared (as well as the interrupt) by writing a 1 to the CTI bit. If the TPIE bit is cleared, this bit is also cleared. 0: no intrusive interrupt; 1: intrusive interrupt generated;
8	TEF	R	0	Intrusion Event Flag. This bit is set by hardware to 1 when an intrusion event is detected. This flag bit can be cleared by writing a 1 to the CTE bit. 0: No intrusion event; 1: Intrusion event detected; Note: An intrusion event resets all BKP_DRx registers. All BKP_DRx registers remain reset as long as TEF is 1. When this bit is set to 1, if a write operation is performed to the BKP_DRx, the value written will not be saved.
7:3	Reserved	RW	-	Reserved
2	TPIE	RW	0	Allow intrusion of TAMPER pin interrupts. 0: Intrusion into the detection interrupt is prohibited; 1: Intrusion to monitor the terminal is allowed (the TPE bit of the BKP_CR register must also be set to 1) Note 1: Intrusion interrupts cannot wake up the system core from low-power mode; Note 2: Reset this bit only after a system reset or wake-up from standby mode.
1	CTI	W	0	Clears the intrusion detection interrupt. This bit can only be written to and read out with a value of 0. 0: Invalid; 1: Clears the Intrusion Detection Interrupt and TIF Intrusion Detection Interrupt flags.
0	CTE	W	0	Clears the intrusion detection event. This bit can only be written to and read out with a value of 0. 0: Invalid; 1: Clears the TEF intrusion detection event flag (and resets the intrusion detector).

6.3.5. BKP register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40006C00	Res	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	BKP_DR1	Res.																D[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	BKP_DR2	Res.																D[15:0]															
	Reset value	0																RW															
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	BKP_DR3	Res.																D[15:0]															
	Reset value	0																RW															
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	BKP_DR4	Res.																D[15:0]															
	Reset value	0																RW															
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	BKP_DR5	Res.																D[15:0]															
	Reset value	0																RW															
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	BKP_DR6	Res.																D[15:0]															
	Reset value	0																RW															
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	BKP_DR7	Res.																D[15:0]															
	Reset value	0																RW															
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	BKP_DR8	Res.																D[15:0]															
	Reset value	0																RW															
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	BKP_DR9	Res.																D[15:0]															
	Reset value	0																RW															
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	BKP_DR10	Res.																D[15:0]															
	Reset value	0																RW															
																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	BKP_DR10	Res.																						ASOS	ASOE	CCO	CAL[6:0]						
	Reset	0																						RW									
																								0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	value																																
0x30	BKP_CR	Res.																														TPAL	TPE
	Reset value	0																														RW	
																																0	0
0x34	BKP_CSR	Res.																			TIF	TEEF	Res.							TPIE	CTI	CTE	
	Reset value	0																			R	R	0							RW	W	W	
0x38	Res.																																
	Reset value	0																															
0x3C	Res.																																
	Reset value	0																															
0x40	BKP_DR11	Res.																D[15:0]															
	Reset value	0																RW															
0x44	BKP_DR12	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x48	BKP_DR	Res.																D[15:0]															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	13																																
	Reset value	0																RW															
																		0															
0x4C	BKP_DR14	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x50	BKP_DR15	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x54	BKP_DR16	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x58	BKP_DR17	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x5c	BKP_DR18	Res.																D[15:0]															
	Reset value	0																RW															
																		0															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	e																																
0x60	BKP_DR19	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x64	BKP_DR20	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x68	BKP_DR21	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x6c	BKP_DR22	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x70	BKP_DR23	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x74	BKP_DR	Res.																D[15:0]															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	24																																
	Reset value	0																RW															
																		0															
0x78	BKP_DR25	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x7C	BKP_DR26	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x80	BKP_DR27	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x84	BKP_DR28	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x88	BKP_DR29	Res.																D[15:0]															
	Reset value	0																RW															
																		0															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	e																																
0x8C	BKP_DR30	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x90	BKP_DR31	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x94	BKP_DR32	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x98	BKP_DR33	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0x9C	BKP_DR34	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0XA0	BKP_DR	Res.																D[15:0]															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	35																																
	Reset value	0																RW															
																		0															
0xA4	BKP_DR36	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0XA8	BKP_DR37	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0xAC	BKP_DR38	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0XB0	BKP_DR39	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0xB4	BKP_DR40	Res.																D[15:0]															
	Reset va	0																RW															
																		0															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	lue																																
0XB8	BKP_DR41	Res.																D[15:0]															
	Reset value	0																RW															
																		0															
0xBC	BKP_DR42	Res.																D[15:0]															
	Reset value	0																RW															
																		0															

7. CRC calculation unit (CRC)

7.1. Introduction

The Cyclic Redundancy Check (CRC) computation unit is based on a fixed generating polynomial to obtain a 32-bit CRC computation.

Among other applications, the CRC technique is mainly used to verify the correctness and integrity of data transmission or data storage.

7.2. Main features of CRC

- Use CRC-32 (Ethernet) polynomials: 0x4C11DB7
  — X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X4 + X2 + X +1
- A 32-bit data register for input/output.
- CRC calculation time: 4 AHB clock cycles (HCLK)
- General purpose 8-bit register (can be used to store temporary data)

7.3. Description of the CRC function

7.3.1. CRC block diagram

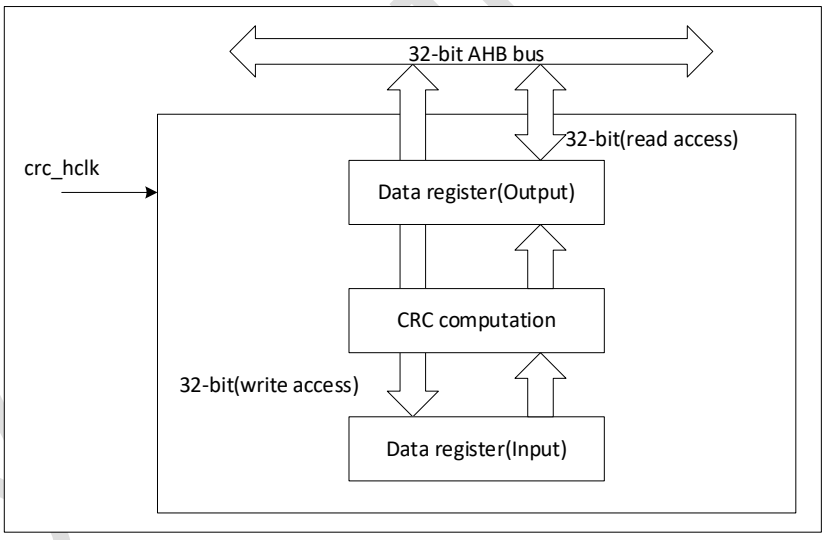


Figure 7-1 Block diagram of the CRC calculation unit

The CRC calculation unit contains one 32-bit data register:

When a write operation is performed on this register, it serves as an input register, allowing new data to be entered for CRC calculation.

A read operation to this register returns the result of the last CRC calculation.

For each write to the data register, the result of the calculation is a combination of the result of the previous CRC calculation and the result of the new calculation (CRC calculation is performed on the entire 32-bit word, not byte by byte).

Register CRC_DR can be reset to 0xFFFF FFFF by setting the RESET bit of register CRC_CR. This operation does not affect the data within register CRC_IDR.

7.4. CRC register

These registers must be accessed in a 32-bit manner.

7.4.1. Data register (CRC_DR)

Address offset:0x00

Reset value:0xFFFF FFFF

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

Bit	Name	R/W	Reset Value	Function
31 : 0	DR	RW	32'hFFFFFFFF	Data register bits. Used as an input register when writing new data to the CRC counter. Returns the result of the CRC calculation when read.

7.4.2. Independent data register (CRC_IDR)

Address offset:0x04

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								IDR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31 : 8	Reserved	RES	-	Reserved
7 : 0	IDR	RW	8'h0	General purpose 8-bit data register bit. Can be used to temporarily store 1 byte of data. The CRC reset generated by the RESET bit of register CRC_CR has no effect on this register. Note: This register does not participate in CRC calculations and can hold any data.

7.4.3. Control register (CRC_CR)

Address offset:0x08

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															RESET
															w

Bit	Name	R/W	Reset Value	Function
31 : 1	Reserved	RES	-	Reserved
0	RESET	W	0	A software set will reset the CRC module and the data register will be loaded with the value of the CRC_INIT register. Software can only write 1, cleared by hardware.

7.4.4. CRC register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET
	Reset																																0

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

8. General-purpose I/O (GPIO)

8.1. Introduction

Each general-purpose I/O port consists of four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 32-bit bit/reset register (GPIOx_BSRR), a 32-bit latch register (GPIOx_LCKR), and two 32-bit multiplexing function selection registers (GPIOx_AFRH and GPIOx_AFRL).

8.2. Main features

- Output state: push-pull or open-drain + pull-up/pull-down
- Output data from output data register (GPIOx_ODR) or peripheral (multiplexed function output)
- Different speeds can be selected for each I/O
- Input states: float, pull-up/pull-down, analogue
- Input data to input data register (GPIOx_IDR) or peripheral (multiplexed function input)
- Set and reset registers (GPIOx_BSRR), with bitwise write access to GPIOx_ODR
- Lockout mechanism (GPIOx_LCKR) to freeze I/O configuration
- Analogue function
- Multiplexed function input/output selection registers (up to 16 multiplexed functions per I/O)
- Fast rollover with as little as two clock cycles per rollover
- Pin multiplexing is very flexible, allowing I/O pins to be used as GPIOs or as one of several peripheral functions.

8.3. Functional descriptions

Each bit of each GPIO can be configured in several modes through software programming:

- Input float
- Input pull-up
- Input pull-down
- Analogue function
- Open-drain output with pull-up or pull-down function
- Push-pull output with pull-up or pull-down function
- Multiplexed push-pull with pull-up or pull-down function
- Open-drain multiplexing with pull-up or pull-down function

Each I/O port bit is freely programmable, but the I/O port registers must be accessed as 32-bit words, half-words, or bytes. The GPIOx_BSRR register is designed to enable atomic read/modify access to the GPIO ODR registers. This ensures that interrupt requests occurring between read and modify accesses will not be a problem.

The following figure shows the basic structure of the 5 V tolerant and basic I/O port bits as well as possible port bit configuration options.

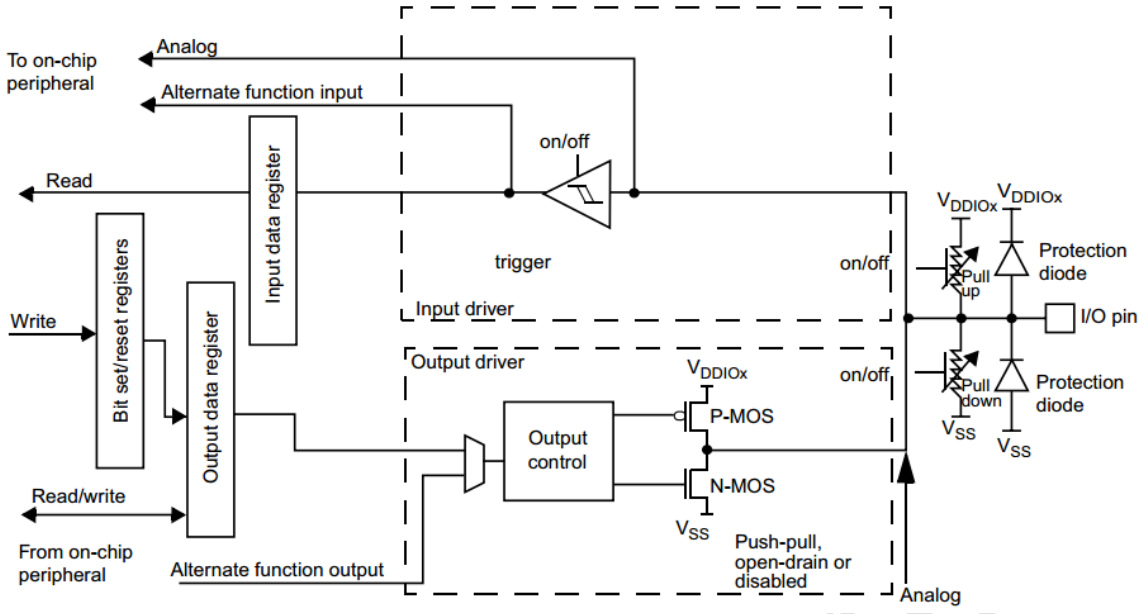


Figure 8-1 Basic structure of common I/O port bits

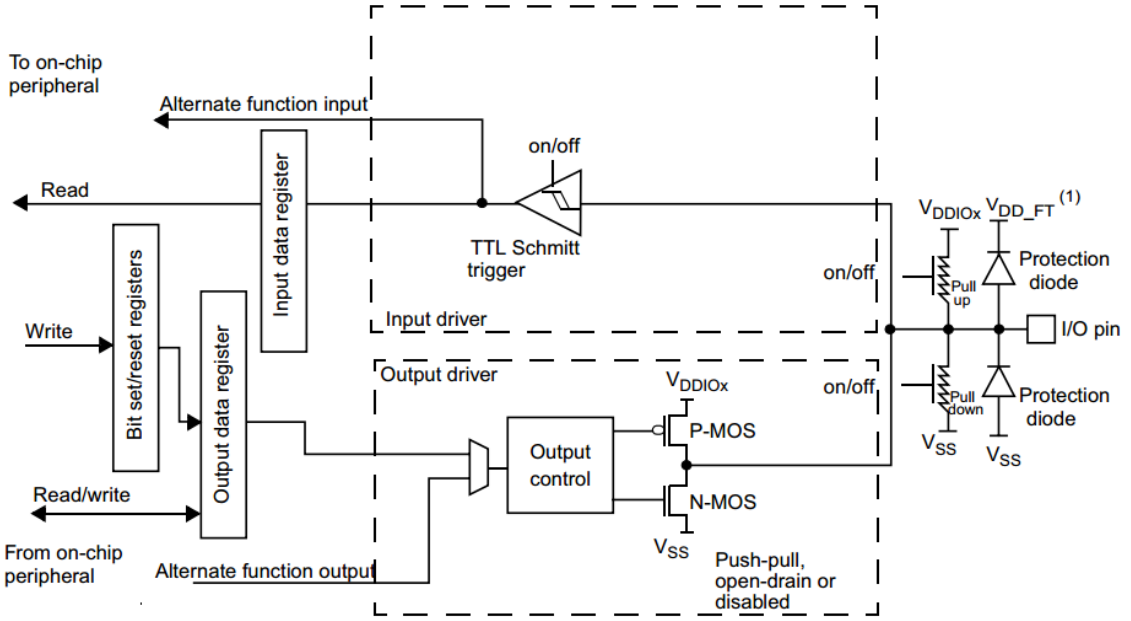


Figure 8-2 5V Tolerance I/O Port Bit Basic Structure

Table 8-1 Port Bit Configuration

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]	PUPD(i)[1:0]		I/O configuration	
01	0	SPEED[1:0]	0	0	GP output	PP
	0		0	1	GP output	PP+PU
	0		1	0	GP output	PP+PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD+PU

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]		PUPD(i)[1:0]		I/O configuration	
	1			1	0	GP output	OD+PD
	1			1	1	Reserved(GP output OD)	
10	0	SPEED[1:0]		0	0	AF	PP
	0			0	1	AF	PP+PU
	0			1	0	AF	PP+PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD+PU
	1			1	0	AF	OD+PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved(input floating)	
11	x	x	x	0	0	Input/Output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/Output	Analog,PD
	x	x	x	1	1	Reserved	

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

8.3.1. General-purpose I/O (GPIO)

During and just after reset, the multiplexing function has not been activated and the I/O ports are configured for input float mode analogue mode. After reset, the debug pins are in a multiplexed function pull-up/pull-down state:

- PA15: JTDI in pull-up state
- PA14: JTCK/SWCLK in pull-down state
- PA13: JTMS/SWDAT in pull-down state (F4 is pull-down, G4 is pull-up, F1 is pull-up)
- PB4: NJTRST is in pull-up state
- PB3: JTDO in float state

When the pin is configured as an output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. The output driver can be used in push-pull mode or in open-drain mode (only N-MOS is active when output is 0).

The input data register (GPIOx_IDR) captures the data from the I/O pins every 1 AHB1 clock cycle.

All GPIO pins have internal weak pull-up and pull-down resistors that can be turned on/off based on the value in the GPIOx_PUPDR register.

8.3.2. I/O Pin Multiplexer and Mapping

The microcontroller I/O pins are connected to the on-board peripherals/modules through a multiplexer that allows only one peripheral's multiplexing function (AF) to be connected to an I/O pin at a time. This ensures that there are no conflicts between peripherals that share the same I/O pin.

Each I/O pin has a multiplexer that uses 16 multiplexed function inputs (AF0 to AF15), which can be configured through the GPIOx_AFRL (for pins 0 to 7) and GPIOx_AFRH (for pins 8 to 15) registers:

- Upon completion of the reset, all I/Os are connected to the system's multiplexing function 0 (AF0). All I/Os are configured for multiplexing via the GPIOx_MODER register.
- Multiplexing function per pin
- In addition to this flexible I/O multiplexing architecture, each peripheral can map multiplexing functions to different I/O pins, which optimises the number of peripherals available in a small package.

To configure the I/O to the desired function, follow the steps below:

- Debug Functions: After each reset, these debug function pins are the multiplexed function pins that are immediately available to the debugger by default
- GPIO: Configure the corresponding I/O port as output, input or analogue mode in GPIOx_MODER.
- Peripheral multiplexing function
  - Connect I/O to the desired AFx in GPIOx_AFRL or GPIOx_AFRH registers
  - Select type, pull-up/down, and output speed via GPIOx_OTYPER, GPIOx_PUPDR, and GPIOx_OSPEEDER registers, respectively
  - Configure the required I/O as multiplexed in the GPIOx_MODER registers
- Additional Functions
  - For ADC, you need to configure the corresponding I/O as analogue mode through register GPIOx_MODER, and configure the corresponding functions in ADC.
  - For the additional functions RTC, WKUPx and external crystal interface, the corresponding functions need to be configured in the RTC and PWR modules. These functions, take precedence over the AF selection registers in the GPIO.

8.3.3. I/O Port Control Register

Each GPIO has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) that can be configured for up to 16 I/Os. The GPIOx_MODER register is used to select the I/O direction (input, output, AF, analogue). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed, respectively (the I/O speed pins are connected directly to the corresponding GPIOx_OSPEEDR register bits, regardless of which I/O direction is used). The GPIOx_PUPDR register is used to select pull-up/pull-down regardless of I/O direction.

8.3.4. I/O Port Data Register

Each GPIO has two 16-bit data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR is used to store the data to be output, which can be accessed

read/write. The data input via I/O is stored into the input data register (GPIOx_IDR), which is a read-only register.

8.3.5. I/O Data Bit Operations

The Set Reset Register (GPIOx_BSRR) is a 32-bit register that allows the application to perform set and reset operations on each individual data bit in the Output Data Register (GPIOx_ODR). The size of the Set Reset Register is twice the size of GPIOx_ODR.

Each data bit in GPIOx_ODR corresponds to two control bits in GPIOx_BSRR: BS (i) and BR (i). When a 1 is written, the BS(i) bit sets the corresponding ODR(i) bit. When a 1 is written, the BR (i) bit clears the corresponding ODR (i) bit.

Writing a 0 to any bit in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If an attempt is made to perform both a set and a clear operation on a bit in GPIOx_BSRR, the set operation takes precedence.

Changing the value of each bit in GPIOx_ODR using the GPIOx_BSRR register is a "one-shot" operation and does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can be accessed directly at any time. The GPIOx_BSRR register provides a way to perform atomic per-bit processing.

Software does not need to disable interrupts when performing bitwise operations on GPIOx_ODR: one or more bits can be modified in a single atomic AHB1 write access.

8.3.6. GPIO Locking Mechanisms

The GPIO control registers can be frozen by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers include GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, and GPIOx_AFRH.

To perform a write operation to the GPIOx_LCKR register, a specific write/read sequence must be applied. When the correct LOCK sequence is applied to hit 16 of this register, the value of LCKR[15:0] is used to lock the I/O configuration (the value of LCKR[15:0] must be the same during the write sequence). After applying a LOCK sequence to a port bit, the value of that port bit cannot be modified until the next reset is performed. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, and GPIOx_AFRH).

The LOCK sequence can only be performed by means of a word (32-bit long) access to the GPIOx_LCKR register, since bit 16 of GPIOx_LCKR must be set at the same time as bits [15:0].

Refer to the GPIOx_LCKR register description for details.

8.3.7. I/O Multiplexed Function Input/Output Mode Configuration

Two registers can be used to select from the 16 multiplexed inputs/outputs available for each I/O. With the help of these registers, a multiplexed function can be connected to one of the other pins according to the requirements of the application.

This means that multiple available peripheral functions can be multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH multiplexing registers. In this way, the application can select any of the available functions for each I/O. Since the AF select signal is shared by the multiplexed function inputs and multiplexed function outputs, only one channel needs to be selected for each I/O's multiplexed function input/output.

Note: For each I/O, the application can only select one available peripheral function at a time.

8.3.8. External interrupt/wake-up lines

All ports have external interrupt capability. To use an external interrupt line, the port must be configured in input mode, see Interrupts and Events.

8.3.9. I/O input configuration

When programming an I/O port as an input:

- output buffer is switched off
- Schmitt trigger input is turned on
- Pull-up and pull-down resistors are turned on or off according to the values in the GPIOx_PUPDR registers
- Input data register samples data on I/O pins every 1 AHB1 clock cycle
- Read access to the input data register to obtain I/O status

8.3.10. Output configuration

When programming an I/O port as an output:

- The output buffer is switched on:
  - Open-drain mode: "0" in the output register activates the N-MOS, while "1" in the output register keeps the port in Hi-Z state (P-MOS is always inactive).
  - Push-Pull mode: "0" in the output register activates N-MOS, while "1" in the output register activates P-MOS.
- Schmitt trigger input is switched on.
- Weak pull-up and pull-down resistors are turned on according to the value in the GPIOx_PUPDR register.
- Input data register samples data on I/O pins every 1 AHB clock cycle
- Read accesses to the input data registers obtain the I/O status
- Read access to the output data register obtains the last write value

8.3.11. Configuration of the multiplexing function

When programming the I/O port as a multiplexing function:

- the output buffers can be configured as open-drain or push-pull
- The output buffer is driven by signals from the periphery (transmitter enable and data)
- Schmitt trigger input is switched on
- Weak pull-up and pull-down resistors are turned on or off based on the value in the GPIOx_PUPDR registers
- Input data register samples data on I/O pins every 1 AHB clock cycle
- A read access to the input data register obtains the I/O state

8.3.12. Analogue configuration

When programming an I/O port as an analogue configuration:

- The output buffer is disabled.

- The Schmitt trigger input is deactivated and the power consumption of each analogue input of the I/O pins becomes zero. The output of the Schmitt trigger is forced to handle a constant value (0).
- The weak pull-up and pull-down resistors are disabled.
- Read access to the input data register is "0".

Note: In analogue configuration, the I/O pins are not 5 V tolerant.

8.3.13. HSE or LSE Pins Configured as GPIO

After system reset, HSE and LSE are off in the default mode, at this time the corresponding pins of HSE and LSE can be used as normal GPIO.

After HSE or LSE is enabled via RCC_CSR register, the corresponding pins can only be configured as crystal oscillator function. The priority is higher than GPIO.

When the crystal is configured as external input clock mode, OSC_IN or OSC32_IN is used as clock input, and OSC_OUT or OSC32_OUT can be used as normal I/Os.

8.3.14. BKP Area GPIO Usage

When the VCCD is powered down, PC13/PC14/PC15 will not have GPIO functionality. If the RTC is not using these pins, these pins are set to analogue mode.

8.4. Register Descriptions

All GPIO-related registers can be accessed via byte (8-bit), half-word (16-bit), or word (32-bit) access to the GPIO registers.

8.4.1. GPIO Port Mode Register (GPIOx_MODER) (x= A..E)

Address offset:0x00

Reset value: 0xABFF_FFFF (portA)

Reset value:0xFFFF_FEBF (portB)

Reset value:0xFFFF FFFF (else)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1		MODER14[1		MODER13[1		MODER12[1		MODER11[1		MODER10[1		MODER9[1		MODER8[1	
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**
MODER7[1:		MODER6[1:		MODER5[1:		MODER4[1:		MODER3[1:		MODER2[1:		MODER1[1		MODER0[1	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31 : 0	MODEy[1:0]	RW		y = 15.0 The software configures the corresponding I/O modes with these bits 00: Input mode

				01: General-purpose output mode 10: Multiplexed function mode 11: Analogue mode (reset state)

8.4.2. GPIO Port Output Type Register (GPIOx_OTYPER) (x= A..E)

Address offset:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OT1	OT1	OT1	OT1	OT1	OT1	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	OT	RW		Software Configuration of I/O Output Types 0: Push-pull output (reset state) 1: Open-drain output

8.4.3. GPIO Port Output Speed Register (GPIOx_OSPEEDR) (x= A..E)

Address offset:0x08

Reset value:0x0C00 0000 (port A)

Reset value:0x0000 0000 (else)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15		OSPEED14		OSPEED13		OSPEED12		OSPEED11		OSPEED10		OSPEED9		OSPEED8	
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**
OSPEED7		OSPEED6		OSPEED5		OSPEED4		OSPEED3		OSPEED2		OSPEED1		OSPEED0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:0	OSPEEDy[1:0]	RW		Y = 15..0 Software configuration of IO port output speed 00: very low 01: low speed 10: High speed 11: very high

8.4.4. GPIO Port Pull-Up/Pull-Down Register (GPIOx_PUPDR) (x= A..E)

Address offset:0x0C

Reset value:0x6400 0000 (port A)

Reset value:0x0000 0100 (port B)

Reset value:0x0000 0000 (else)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0		PUPD14[1:0		PUPD13[1:0		PUPD12[1:0		PUPD11[1:0		PUPD10[1:0		PUPD9[1:0		PUPD8[1:0	
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**
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0		PUPD0[1:0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:0	PUPDy [1:0]	RW		Y = 15..0 Software configuration of I/O port pull-up or pull-down 00: No pull-up or pull-down 01: Pull-up 10: Pull-down 11: Reserved

8.4.5. GPIO Port Input Data Register (GPIOx_IDR) (x= A..E)

Address offset:0x10

Reset value:0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	IDy	R		Y = 15..0 This is read-only, and the readout value bit corresponds to the status of the I/O port.

8.4.6. GPIO Port Output Data Register (GPIOx_ODR) (x= A..E)

Address offset:0x14

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**
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved			
15: 0	ODy[1:0]	RW		y = 15..0 Software readable and writable. Note: For GPIOx_BSRR or GPIOx_BRR registers.(x=A,B,C,D,E), each ODR bit can be set/cleared independently.

8.4.7. GPIO Port Reset/Reset Register (GPIOx_BSRR) (x= A..E)

Address offset:0x18

Reset value:0x0000 0000

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
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**
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Name	R/W	Reset Value	Function
31:16	BRy	W		y = 15..0 Software writable, read out return value is 0 0: no effect on the corresponding ODRy bits 1: Clear the corresponding ODRy bit Note: If the corresponding bits of Bsy and Bry are set at the same time, the Bsy bit works.
15: 0	BSy	W		y = 15..0 Software writable, return value is 0 when read. 0: No effect on the corresponding ODRy bit. 1: Set the corresponding ODRy bit

8.4.8. GPIO Port Configuration Lock Register (GPIOx_LCKR) (x= A..E)

This register is used to lock the configuration of the port bits when bit16 (LCKR) is set by executing the correct write sequence. bit [15:0] is used to lock the configuration of the GPIO port. LCKR [15:0] cannot be changed during a defined write operation. After the LOCK sequence has been performed on the corresponding port, the configuration of the port bits will not be able to be changed again until the next system reset.

Note: The special write timing sequence is used to write the GPIOx_LCKR register. Only word accesses can be performed during the lock sequence.

Each latched bit freezes a specific type of configuration register (control and multiplexing function registers)

Address offset:0x1C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															LCK16
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16	LCKK	RW		This bit can be read out at any time, it can only be modified by a lock-key write sequence 0: Port configuration lock bit not activated 1: Port configuration lock key bit is activated and GPIOx_LCKR register is locked before the next system reset. LOCK key write sequence. LOCK key write sequence: Write 1->Write 0->Write 1->Read 0->Read 1, the last read can be omitted, but it can be used to confirm the lock key is activated. Note: The value of LCK [15:0] cannot be changed during the operation of the lock key write timing. Any error in the lock key timing will terminate the lock key being activated. After the first latch sequence on any bit of the port, reading the LCKK bit returns a 1, which results in a direct MCU reset or a peripheral reset.
15: 0	LCKy	RW		y = 15..0 These bits are readable and writable but can only be written if the LCKK bit is 0. 0: Unlocked port configuration 1: Locked port configuration

8.4.9. GPIO Multiplexing Function Low Register (GPIOx_AFRL) (x= A..E)

Address offset:0x20

Reset value:0x0000 0000

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

AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
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**
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0] ((y= 7 to 0))	RW		Software Writable These Bit Configuration Multiplexing Function I/Os AFSELy options. 0000:AF0 1000: AF8 0001:AF1 1001: AF9 0010:AF2 1010: AF10 0011:AF3 1011: AF11 0100:AF4 1100: AF12 0101:AF5 1101: AF13 0110:AF6 1110: AF14 0111:AF7 1111: AF15

8.4.10. GPIO Multiplexing High Register (GPIOx_AFRH) (x= A..E)

Address offset:0x24

Reset value:0x0000 0000

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
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**
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:0	AFSELy[3:0] ((y= 8 to 15))	RW		Software Writable These Bit Configuration Multiplexing Function I/Os AFSELy options. 0000:AF0 1000: AF8 0001:AF1 1001: AF9 0010:AF2 1010: AF10 0011:AF3 1011: AF11 0100:AF4 1100: AF12 0101:AF5 1101: AF13 0110:AF6 1110: AF14 0111:AF7 1111: AF15

8.4.11. GPIO Port Bit Reset Register (GPIOx_BRR) (x= A..E)

Address offset:0x28

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	2	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	Bry	RW		y = 15..0 These bits are writable by the software and return 0 when read. 0: no effect on the corresponding Ody bits 1: Clear the corresponding Ody bit

8.4.12. GPIO register address map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	GPIOB_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	GPIOF_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04	GPIOF_OTYPER (x=A, B, F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOA_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOx_OSPEEDR (x=B, F)	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOA_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOB_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	value																																
0x0C	GPIOF_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (x= A, B, F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	GPIOx_ODR (x= A, B, F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (x= A, B, F)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	(x= A, B, F)																	1	1	1	1	1	1										
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOx_AFRL (x= A, B, F)	AFSEL7 [3:0]				AFSEL6 [3:0]				AFSEL5 [3:0]				AFSEL4 [3:0]				AFSEL3 [3:0]				AFSEL2 [3:0]				AFSEL1 [3:0]				AFSEL0 [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (x= A, B, F)	AFSEL7 [3:0]				AFSEL6 [3:0]				AFSEL5 [3:0]				AFSEL4 [3:0]				AFSEL3 [3:0]				AFSEL2 [3:0]				AFSEL1 [3:0]				AFSEL0 [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOx_BRR (x= A, B, F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9. System configuration controller (SYSCFG)

9.1. Overview

The SYSCFG module mainly performs the following functions:

- I2C type IO noise filter control enable
- All IO noise filter control enable
- EXTI IO select control
- Mapping of initial programme area according to different boot modes
- DMA peripheral passband select control
- TIMERs breakin and ETR control

9.2. SYSCFG register (baseaddr=0x40010000)

9.2.1. SYSCFG configuration register 1 (SYSCFG_CFGR1)

This register is used as a memory and DMA request remap and to control the specific configuration of special IO functions.

Two bits are used to configure the type of access to memory address 0x0000 0000. These two bits are used to select the software physical remap and bypass the hardware BOOT selection. After a reset, these bits use the values configured by the actual BOOT mode.

Address offset：0x00

Reset value：0x0000 000x(x is the memory mode selected by the actual boot mode configuration)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.					Res.	Res.	I2C_PB12	I2C_PB11	I2C_PB10	I2C_PB9	I2C_PB8	I2C_PB7	I2C_PB6	I2C_PB5
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**
Res.														MEM_MODE[1:0]	
														RW	

Bit	Name	R/W	Reset Value	Function
31:24	Reserved	RW	-	readable and writable
23: 16	I2C_IOx_EIIC	RW	0	Analogue filter enable control for I2C related IOs PB12~PB5 0: Analogue filter off 1: Analogue filter enable
15：2	Reserved	RW		readable and writable

Bit	Name	R/W	Reset Value	Function
1:0	MEM_MODE [1:0]	RW	0	Memory mapping selection bit Software set and software clear. They control the mapping of the 0x0000 0000 address of the memory. after a reset, these bits use the actual real boot mode configuration values. 00：Main flash, mapped t 0x0000 0000 01：System flash , mapped at 0x0000 0000 10：ESMC，mapped at 0x0000 0000 11：SRAM, mapped at 0x0000 0000

9.2.2. SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset：0x04

Reset value：0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	Res.	Res.	Res.	Res.	Res.	PVD_LOCK	Res.	LOCKUP_LOCK
													RW		RW

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	-	-	Reserved
11	ADC2_ETRGREG_REMAP	RW	0	ADC2 Rule Conversion External Trigger Remap This bit can be set '1' or set '0' by software. It controls the trigger input that is connected to the ADC2 Rule Conversion External Trigger. When this bit is '0', the ADC2 rule-conversion external trigger is connected to EXTI11; when this bit is '1', the ADC2 rule-conversion external trigger is connected to TIM8_TRGO.

Bit	Name	R/W	Reset Value	Function
10	ADC2_ETRGINJ_REMAP	RW	0	ADC2 Injection Conversion External Trigger Remap This bit can be set '1' or '0' by software. It controls the trigger input that is connected to the ADC2 Injection Conversion External Trigger. When this bit is set to '0', the ADC2 injection conversion external trigger is connected to EXTI15; when this bit is set to '1', the ADC2 injection conversion external trigger is connected to TIM8 channel 4.
9	ADC1_ETRGREG_REMAP	RW	0	ADC1 Rule Conversion External Trigger Remap This bit can be set '1' or '0' by software. It controls the trigger input that is connected to the ADC1 rule-conversion external trigger. When this bit is '0', the ADC1 rule-conversion external trigger is connected to EXTI11; when this bit is '1', the ADC1 rule-conversion external trigger is connected to TIM8_TRGO.
8	ADC1_ETRGINJ_REMAP	RW	0	ADC1 Injection Conversion External Trigger Remap This bit can be set '1' or '0' by software. It controls the trigger input that is connected to the ADC1 injection conversion external trigger. When this bit is '0', the ADC1 injection external trigger is connected to EXTI15; when this bit is '1', the ADC1 injection external trigger is connected to TIM8 channel 4.
7:3	Reserved	-	-	Reserved
2	PVD_LOCK	RW	0	PVD Lock Enable Bit Set by software and cleared by system reset. It can be used to enable and lock the PVD connection to TIM1's brake inputs, and also to lock PVDE in the PWR_CR register. 0: The PVD interrupt is not connected to the brake input of TIM1. the PVDE bit can be written by the application. 1: The PVD interrupt is connected to the brake input of TIM1. the PVDE bit is read-only.
1	Reserved	-	-	Reserved
0	LOCKUP_ LOCK	RW		Cortex-M4F LOCKUP bit enable bit Set by software and cleared by system reset. It enables and locks the LOCKUP (hardfault) output of the Cortex-M4F to the brake inputs of TIM1 and TIM8. 0: The LOCKUP output of the Cortex-M4F is not connected to the brake inputs of TIM1 and TIM8; 1: The LOCKUP output of the Cortex-M4F is connected to the brake inputs of TIM1 and TIM8;

9.2.3. SYSCFG configuration register 3 (SYSCFG_CFGR3)

Address offset：0x08

Reset value：0x7F7F_7F7F

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

Res.	DMA4_MAP							Res.	DMA3_MAP						
	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**
Res.	DMA2_MAP							Res.	DMA1_MAP						
	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Reserved
30:24	DMA4_MAP	RW	7'h7f	DMA1 channel 4 mapping. Refer to the DMA1_MAP description.
23	Res	-	-	Reserved
22:16	DMA3_MAP	RW	7'h7f	DMA1 channel 3 mapping. Refer to the DMA1_MAP description.。
15	Res	-	-	Reserved
14:8	DMA2_MAP	RW	7'h7f	DMA1 channel 2 mapping. Refer to the DMA1_MAP description.
7	Res	-	-	Reserved
6:0	DMA1_MAP	RW	7'h7f	DMA1 channel 1 mapping. 0000000: ADC1; 0000001: ADC2; 0000010: ADC3; 0000011: Reserved; 0000100: Reserved; 0000101: SPI1_RD; 0000110: SPI1_WR; 0000111: SPI2_RD; 0001000: SPI2_WR; 0001001: SPI3_RD; 0001010: SPI3_WR; 0001011: USART1_RD; 0001100: USART1_WR; 0001101: USART2_RD; 0001110: USART2_WR; 0001111: USART3_RD; 0010000: USART3_WR; 0010001: USART4_RD; 0010010: USART4_WR; 0010011: USART5_RD; 0010100: USART5_WR; 0010101: I2C1_RD; 0010110: I2C1_WR; 0010111: I2C2_RD; 0011000: I2C2_WR; 0011001: TIM1_CH1; 0011010: TIM1_CH2;

Bit	Name	R/W	Reset Value	Function
				0011011: TIM1_CH3; 0011100: TIM1_CH4; 0011101: TIM1_COM; 0011110: TIM1_TRG; 0011111: TIM1_UP; 0100000: TIM2_CH1; 0100001: TIM2_CH2; 0100010: TIM2_CH3; 0100011: TIM2_CH4; 0100100: TIM2_UP; 0100101: TIM3_CH1; 0100110: TIM3_CH3; 0100111: TIM3_CH4; 0101000: TIM3_UP; 0101001: TIM3_TRIG; 0101010: TIM4_CH1; 0101011: TIM4_CH2; 0101100: TIM4_CH3; 0101101: TIM4_UP; 0101110: TIM5_CH1; 0101111: TIM5_CH2; 0110000: TIM5_CH3; 0110001: TIM5_CH4; 0110010: TIM5_UP; 0110011: TIM5_TRIG; 0110100: TIM6; 0110101: TIM7; 0110110: TIM8_CH1; 0110111: TIM8_CH2; 0111000: TIM8_CH3; 0111001: TIM8_CH4; 0111010: TIM8_COM; 0111011: TIM8_TRG; 0111100: TIM8_UP; 0111101:TIM2_TRIG; 0111110: TIM3_CH2; 0111111: TIM4_CH4; 1000000: TIM4_TRIG; 1000001: ESMC_TX; 1000010: ESMC_RX; 1000011: SDIO; 1000100: USB； Others：Reserved

9.2.4. SYSCFG configuration register 4 (SYSCFG_CFGR4)

Address offset：0x0C

Reset value：0x7F7F_7F7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DMA8_MAP							Res.	DMA7_MAP						
	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**
Res.	DMA6_MAP							Res.	DMA5_MAP						
	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Reserved
30:24	DMA4_MAP	RW	7'h7f	DMA2 channel 1 mapping. See DMA1_MAP description.
23	Res	-	-	Reserved
22:16	DMA7_MAP	RW	7'h7f	DMA2 channel 7 mapping. See DMA1_MAP description.
15	Res	-	-	Reserved
14:8	DMA6_MAP	RW	7'h7f	DMA2 channel 6 mapping. See DMA1_MAP description.
7	Res	-	-	Reserved
6:0	DMA5_MAP	RW	7'h7f	DMA2 channel 5 mapping. See DMA1_MAP description.

9.2.5. SYSCFG configuration register 5 (SYSCFG_CFGR5)

Address offset：0x10

Reset value：0x7F7F_7F7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DMA12_MAP							Res.	DMA11_MAP						
	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**
Res.	DMA10_MAP							Res.	DMA9_MAP						
	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Reserved
30:24	DMA12_MAP	RW	7'h7f	DMA2 channel 5 mapping. See DMA1_MAP description.
23	Res	-	-	Reserved
22:16	DMA11_MAP	RW	7'h7f	DMA2 channel 4 mapping.

Bit	Name	R/W	Reset Value	Function
				See DMA1_MAP description.
15	Res	-	-	Reserved
14:8	DMA10_MAP	RW	7'h7f	DMA2 channel 3 mapping. See DMA1_MAP description.
7	Res	-	-	Reserved
6:0	DMA9_MAP	RW	7'h7f	DMA2 channel 2 mapping. See DMA1_MAP description.

9.2.6. External interrupt configuration register 1 (SYS_EXTICR1)

Address offset:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	EXTIx[3:0]	RW	0x0000	EXTIx configuration (x = 0 ... 3) (EXTI x configuration) These bits can be read or written by software to select the input source for the EXTIx external interrupt. Reference EXTI external interrupt event mapping 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin

9.2.7. External Interrupt Configuration Register 2 (SYS_EXTICR2)

Address offset:0x18

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			

Bit	Name	R/W	Reset Value	Function
15:0	EXTIx[3:0]	RW	0x0000	EXTIx configuration (x =4 ... 7) (EXTI x configuration) These bits can be read and written by software to select the input source for the EXTIx external interrupt. Refer to EXTI external interrupt event mapping. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin

9.2.8. External Interrupt Configuration Register 3 (SYS_EXTICR3)

Address offset:0x1C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	EXTIx[3:0]	RW	0x0000	EXTIx configuration (x = 8 ... 11) (EXTI x configuration) These bits can be read or written by software to select the input source for the EXTIx external interrupt. Refer to EXTI external interrupt event mapping. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin

9.2.9. External Interrupt Configuration Register 4 (SYS_EXTICR4)

Address offset:0x20

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	EXTIx[3:0]	RW	0x0000	EXTIx configuration (x = 12 ... 15) (EXTI x configuration) These bits can be read and written by software to select the input source for the EXTIx external interrupt. Refer to EXTI external interrupt event mapping. 0000: PA[x] pin 0001: PB[x] pin 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin

9.2.10. GPIOA Filter Enable (PA_ENS)

Address offset:0x24

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	PA_ENS[x]	RW	0x0000	Noise filter enable, active high 0: Noise filter bypass 1: Enable noise filter

9.2.11. GPIOB Filter Enable (PB_ENS)

Address offset:0x28

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	PA_ENS[x]	RW	0x0000	Noise filter enable, active high 0: Noise filter bypass 1: Enable noise filter

9.2.12. GPIOC filter enable (PC_ENS)

Address offset:0x2C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	PC_ENS[x]	RW	0x0000	Noise filter enable, active high 0: Noise filter bypass 1: Enable noise filter

9.2.13. GPIOD filter enable (PD_ENS)

Address offset:0x30

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	PD_ENS[x]	RW	0x0000	Noise filter enable, active high 0: Noise filter bypass 1: Enable noise filter

9.2.14. GPIOE Filter Enable (PE_ENS)

Address offset:0x34

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	-	-
15:0	PE_ENS[x]	RW	0x0000	Noise filter enable, active high 0: Noise filter bypass 1: Enable noise filter

9.2.15. GPIO analogue channel enable (GPIO_ENA)

Address offset:0x38

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.										PC_ENA					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.						PB_ENA		PA_ENA							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	-	-	-
21:16	PC_ENA	RW	0x00	PC[5：0] Analogue input enable, active high
15:10	Reserved	-	-	-
9:8	PB_ENA	RW	0x00	PB[1:0] Analogue input enable, active high
7:0	PA_ENA	RW	0x00	PA[7:0] Analogue input enable, active high

9.2.16. Timer clock extension control (TIM_CLK_EXT)

Address offset:0x17C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Tim_pclk2_sel	Tim_pclk1_sel	Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bit	Name	R/W	Reset Value	Function
31:8	Reserved	-	-	-
7	Tim_pclk2 sel	RW	0	Selecting the Timer clock under APB2 bus 0: APB2 bus Timer uses 2x APB2 clock (equal to APB2 clock when APB2 is not divided) 1: APB2 bus Timer always uses APB2 clock.

6	Tim_pclk1 sel	RW	0	Select the Timer clock under APB1 bus 0: APB1 bus Timer uses 2x APB1 clock (equal to APB1 clock when APB1 is not divided) 1: APB1 bus Timer always uses the APB1 clock.
7:0	Reserved	-	-	-

9.2.17. SYSCFG register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SYSCFG_CFGR1	Res.								I2C_PB12	I2C_PB11	I2C_PB10	I2C_PB9	I2C_PB8	I2C_PB7	I2C_PB6	I2C_PB5	Res.														MEM_MODE	
	Read/Write									rw	rw	rw	rw	rw	rw	rw	rw															rw	rw
	Reset Value	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SYSCFG_CFGR2	Res.																													PVD_LOCK	Res.	LOCKUP_LOCK
	Read/Write																														rw	rw	rw
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x08	SYSCFG_CFGR3	Reserved	DMA4_MAP							Reserved	DMA3_MAP							Reserved	DMA2_MAP							Reserved	DMA1_MAP						
	Read/Write		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
	Reset	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Value																																
0x0C	SYSCFG_CFGR4	Reserved	DMA8_MAP							Reserved	DMA7_MAP							Reserved	DMA6_MAP							Reserved	DMA5_MAP						
	Read/Write		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
	Reset Value	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0x10	SYSCFG_CFGR5	Reserved	DMA12_MAP							Reserved	DMA11_MAP							Reserved	DMA10_MAP							Reserved	DMA9_MAP						
	Read/Write		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
	Reset Value	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0x14	SYSCFG_EXTICR1	Res.																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Read/Write																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SYSCFG_EXTI	Res.																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CR2																																
	Read/Write																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SYSCFG_EXTICR3	Res.																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Read/Write																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	SYSCFG_EXTICR4	Res.																EXTI5[3:0]				EXTI4[3:0]				EXTI3[3:0]				EXTI2[3:0]			
	Read/Write																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	PA_ENS	Res.																PA_ENS															
	Read/Write																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Value																																
0x28	PA_ENS	Res.																PB_ENS															
	Read/Write																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	PA_ENS	Res.																PC_ENS															
	Read/Write																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	PA_ENS	Res.																PD_ENS															
	Read/Write																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	PA_ENS	Res.																PE_ENS															
	Read/Write																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	GPIO_ENA	Res.										PC_ENA						Res.						PB_ENA		PA_ENA							

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read/Write											rw	rw	rw	rw	rw	rw							rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x17C	TIM_CLK_EXT	Res.																Res.								Tim_pclk2 sel	Tim_pclk1 sel	Res.					
	Read/Write																									rw	rw						
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10. DMA controller (DMA)

10.1. Overview

Direct memory access (DMA) is used to provide high-speed data transfers between peripherals and memory or between memory and memory. Moving data does not require CPU intervention; data can be moved quickly through DMA, which saves CPU resources for other operations. The two DMA controllers have 12 channels (7 channels for DMA1 and 5 channels for DMA2), each dedicated to managing requests for memory access from one or more peripherals. There is also an arbiter to coordinate the priority of individual DMA requests.

10.1.1. Main features

The main functions are as follows:

- Single AHB master
- Supports peripheral to memory, memory to peripheral, memory to memory and peripheral to peripheral data transfers
- On-chip memory devices such as FLASH, SRAM, AHB and APB peripherals as source and target
- All DMA channels are independently configurable:
  - Each channel is either associated with a DMA request signal from a peripheral or with a software trigger in a memory-to-memory transfer. This configuration is done by software.
  - The priority between requests is programmable by software (4 levels per channel: very high, high, medium, low) and in equal cases by hardware (e.g. requests to channel 1 have priority over requests to channel 2).
  - Source and destination transmission sizes are independent (byte, halfword, word), analogue packing and unpacking. Source and destination addresses must be aligned by data size.
  - Number of programmable transfer data:0 ~65535
- One interrupt request is generated per channel. Each interrupt request is caused by any of three DMA events: transfer complete, half transfer, or transfer error.

10.2. Functional descriptions

10.2.1. DMA transmission

DMA block transfers can be requested from a peripheral or triggered by software in the case of memory-to-memory transfers.

After the event, the individual DMA transfer steps.

- The peripheral sends a DMA request signal to the DMA controller.
- The DMA controller responds to the request based on the priority of the channel associated with this peripheral request.
- Once the DMA controller grants the peripheral, the transfer is initiated and when the transfer is complete, the DMA controller sends an answer to the peripheral.

- When the peripheral gets an answer from the DMA controller, it releases its request.
- Once the peripheral cancels the request assertion, the DMA controller releases the channel request currently occupied by the peripheral.

The request/acknowledgement protocol is used when the peripheral is the source or destination of the transfer. For example, in a memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the DMA controller, which then reads a single piece of data from memory and writes that data to the peripheral.

For a given channel x, a DMA block transfer consists of the following repeating sequence.

- A single DMA transfer encapsulating a single data from two AHB transfers over the DMA AHB master bus:.
  - A single piece of data (byte, half-word, or word) read from a peripheral data register or a location in memory, addressed via the internal current peripheral/memory address register.

    The starting address used for the first single transfer is the base address of the peripheral or memory, configured in the DMA_CPARx or DMA_CMARx registers.
  - A single data write (byte, halfword, or word) to a location in the peripheral data register or memory is addressed through the internal current peripheral/memory address register.

    The starting address used for the first transfer is the base address of the peripheral or memory and is configured in the DMA_CPARx or DMA_CMARx registers.
  - The DMA_CNDTRx register contains the number of data items remaining to be transferred (the number of AHB 'read-after-write' transfers).

Repeat the above sequence until DMA_CNDTRx is empty.

Note: The AHB master bus source/destination address must be aligned with the programme size of the individual data transferred to the source/destination.

10.2.2. Arbiter

The arbiter initiates peripheral/memory accesses based on the priority of the channel request.

When a requesting channel x is granted (hardware-requested or software-triggered) access by the arbiter, a single DMA transfer is issued (e.g., an AHB "read-and-write" transfer of a single piece of data). The arbiter then arbitrates all requested channels again and selects the channel with the highest priority.

Priority is managed in 2 stages:

- Software: the priority of each channel can be set in the DMA_CCRx register and has 4 levels:
  - highest priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: If 2 requests have the same software priority, the lower numbered channel has a higher priority than the higher numbered channel. For example, channel 2 has priority over channel 4.

10.2.3. DMA Channel

Each channel can perform DMA transfers between peripheral registers with fixed addresses and memory addresses. The DMA transfer data amount is programmable up to 65535 bytes, and this register value is decremented after each data transfer.

10.2.3.1. Programmable transfer data amount

The peripheral and memory lump sum transfer data widths are programmable through the PSIZE and MSIZE bits in the DMA_CCRx register.

10.2.3.2. Address Pointer Increment

Peripheral and memory pointer auto-incrementation can be done selectively after each transfer by setting the PINC and MINC flag bits in the DMA_CCRx register.

When set to increment mode, the next address to be transferred will be the previous address plus the increment value, which depends on the selected data width of 1, 2, or 4. The first address to be transferred is the address stored in the DMA_CPARx/DMA_CMARx registers. These registers maintain their initial values during the transfer and software cannot change or read out the address currently being transferred (it is in the internal current peripheral/memory address register).

When the channel is configured in acyclic mode, no further DMA operations will be generated after the transfer has ended (i.e., the transfer count becomes 0). To start a new DMA transfer, the transfer count needs to be rewritten in the DMA_CNDTRx register with the DMA channel closed.

In cyclic mode, at the end of the last transfer, the contents of the DMA_CNDTRx register are automatically reloaded to their initial values and the internal current peripheral/memory address registers are reloaded to the initial base address set by the DMA_CPARx/DMA_CMARx registers.

10.2.3.3. Channel Configuration Procedure

Configure the DMA channel on peripheral request as follows:

- Set the address of the peripheral register in the DMA_CPARx register. When a peripheral data transfer request occurs, this address will be the source or destination of the data transfer.
- Set the address of the data memory in the DMA_CMARx register. When a peripheral data transfer request occurs, the transferred data will be read from or written to this address.
- Set the amount of data to be transferred in the DMA_CNDTRx register. This value is decremented after each data transfer.
- Configure the DMA_CCRx register with the following parameters:
  - Priority of the channel.
  - Direction of data transfer
  - Cyclic mode
  - Peripheral and memory incremental mode
  - Peripheral and memory data size
  - Interrupt enable
- Setting the ENABLE bit of the DMA_CCRx register starts the channel.

Once the DMA channel is activated, it can respond to DMA requests from peripherals connected to the channel.

After half of the data has been transferred, the Half Transfer Flag (HTIF) is set to 1. When the Half Transfer Interrupt Allowed bit (HTIE) is set, an interrupt request is generated. After the end of the data transfer, the Transfer Completion Flag (TCIF) is set to 1. When the Allow Transfer Completion Interrupt Bit (TCIE) is set, an interrupt request is generated.

10.2.3.4. Channel status and disabled channels

An active channel x is an enabled channel (read DMA_CCRx.EN = 1). An active channel x is a channel that must have been enabled by the software (DMA_CCRx. EN = 1) and then no transmission error has occurred (DMA_ISR.TEIFx = 0). If a transmission error occurs, the channel is automatically disabled by hardware (DMA_CCRx.EN = 0).

The following 3 scenarios may occur:

- Suspend and resume the channel

This corresponds to the following two actions.

- The active channel is disabled by software (write DMA_CCRx.EN = 0).
- Software re-enables the channel (DMA_CCRx.EN = 1) but does not reconfigure the other channel registers (e.g., DMA_CNDTRx, DMA_CPARx, and DMA_CMARx); or incomplete transfers hang the bus when software disables it.

The DMA hardware does not support this situation and therefore cannot guarantee that the remaining data transfers will be performed correctly.

- Stopping and Aborting a Channel

If the channel is no longer needed by the application, the active channel can be disabled by software. The channel is stopped and aborted, but the DMA_CNDTRx register contents may not correctly reflect the remaining data transfers.

- Abort and restart the channel

This corresponds to the software sequence: Disable the active channel, then reconfigure the channel and enable it again.

- Hardware support when the following conditions are met.

- The application guarantees that there are no ongoing (not yet completed) transfers in the DMA when the channel is disabled by software. For example, the application can first disable a peripheral in DMA mode to ensure that there are no pending hardware DMA requests for that peripheral.
- The software must perform independent write accesses to the same DMA_CCRx register: first to disable the channel, second to reconfigure the channel for the next block transfer if a configuration change is required, including DMA_CCRx. Finally, to enable the channel again.

When a channel transfer error occurs, hardware clears the EN bit in the DMA_CCRx register. This EN bit cannot be set again by software to reactivate channel x until the TEIFx bit of the DMA_ISR register is set.

10.2.3.5. DMA cycle mode

Cyclic mode is used to handle circular buffers and continuous data transfers (e.g., scan mode for ADCs). The CIRC bit in the DMA_CCRx register is used to enable this function. When cyclic mode is

activated and the number of data transfers becomes 0, it will automatically be restored to the initial value set when the channel was configured and the DMA operation will continue.

Note: Cyclic mode cannot be used in memory-to-memory mode. Software must clear the MEM2MEM bit of the DMA_CCRx register before cyclic mode (CIRC = 1) enables the channel. When cyclic mode is enabled, the amount of data to be transferred will be automatically reloaded using the programmed initial values during the channel configuration phase and DMA requests will continue to be answered.

In order to stop cyclic transfers, software needs to stop the peripheral from generating DMA requests (e.g. exit ADC scan mode) before disabling the DMA channel.

Software must explicitly program the DMA_CNDTRx value before starting/enabling a transfer and after stopping a cyclic transfer.

10.2.3.6. memory-to-memory mode

The operation of the DMA channel can be performed without a peripheral request; this operation is the memory-to-memory mode. When the DMA channel is initiated by software setting the EN bit in the DMA_CCRx register after setting the MEM2MEM bit in the DMA_CCRx register, the DMA transfer will start immediately. The DMA transfer ends when the DMA_CNDTRx register becomes zero.

Memory-to-memory mode cannot be used in conjunction with cyclic mode.

10.2.3.7. Peripheral-to-peripheral mode

Any DMA channel can be operated in peripheral-to-peripheral mode:.

- When a hardware request from a peripheral is selected to trigger a DMA channel

This peripheral is the DMA initiator and transfers data between this peripheral and registers belonging to another memory-mapped peripheral (which is not configured for DMA mode).

- When no peripheral request is selected and connected to a DMA channel

The software configures register-to-register transfers by setting the MEM2MEM bit of the DMA_CCRx register.

10.2.3.8. Configure the transfer direction, specify source/destination

The value of the DIR bit of the DMA_CCRx register sets the direction of the transfer and therefore identifies the source and target, regardless of the source/target type (peripheral or memory):.

- DIR = 1 typically defines a memory-to-peripheral transfer. More generally, if DIR = 1.
  - the source attribute is defined by the DMA_MARx register, the MSIZE[1:0] field, and the MINC bit of the DMA_CCRx register. Regardless of their common nomenclature, these "memory" registers, fields and bits are used to define the source peripheral in peripheral-to-peripheral mode.
  - The target attributes are defined by the DMA_PARx register, the PSIZE[1:0] fields of the DMA_CCRx register, and the PINC bit. Regardless of their usual names, these "peripheral" registers, fields and bits are used to define the target memory in memory-to-memory mode.
- DIR = 0 typically defines a peripheral to memory transfer. More generally, if DIR = 0.

- The source attribute is defined by the DMA_PARx register, the PSIZE[1:0] fields of the DMA_CCRx register, and the PINC bit. Regardless of their common nomenclature, these "peripheral" registers, fields and bits are used to define the source memory in memory-to-memory mode.
- The target attributes are defined by the DMA_MARx register, the MSIZE[1:0] fields of the DMA_CCRx register, and the MINC bit. Regardless of their common designation, these "memory" registers, fields and bits are used to define the target peripheral in peripheral-to-peripheral mode.

10.2.4. Data transfer width/alignment/size end

When the memory data width MSIZE and the peripheral data width PSIZE are different, the DMA aligns the data according to the following table:

Table 10-1 Data alignment

Source width	Target width	transmission number	Source: Address/Data	transmission operation	Target: address/data
8	8	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: Read B0[7:0] at 0x0 and write B0[7:0] at 0x0 2: Read B1[7:0] at 0x1 and write B1[7:0] at 0x1 3: Read B2[7:0] at 0x2 and write B2[7:0] at 0x2 4: Read B3[7:0] at 0x3 and write B3[7:0] at 0x3	0x0/B0 0x1/B1 0x2/B2 0x3/B3
8	16	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: Read B0[7:0] at 0x0 and write 00B0[7:0] at 0x0 2: Read B1[7:0] at 0x1 and write 00B1[7:0] at 0x2 3: Read B2[7:0] at 0x2 and write 00B2[7:0] at 0x4 4: Read B3[7:0] at 0x3 and write 00B3[7:0] at 0x6	0x0/00B0 0x2/00B1 0x4/00B2 0x6/00B3
8	32	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: Read B0[7:0] at 0x0 and write 000000B0[31:0] at 0x0 2: Read B1[7:0] at 0x1 and write 000000B1[31:0] at 0x4 3: Read B2[7:0] at 0x2 and write 000000B2[31:0] at 0x8 4: Read B3[7:0] at 0x3 and write 000000B3[31:0] at 0xC.	0x0/000000B0 0x4/000000B1 0x8/000000B2 0xC/000000B3
16	8	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: Read B1B0[15:0] at 0x0 and write B0[7:0] at 0x0 2: Read B3B2[15:0] at 0x2 and write B2[7:0] at 0x1 3: Read B5B4[15:0] at 0x4 and write B4[7:0] at 0x2	0x0/B0 0x1/B2 0x2/B4 0x3/B6

				4: Read B7B6[15:0] at 0x6 and write B6[7:0] at 0x3	
16	16	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: Read B1B0[15:0] at 0x0 and Write B1B0[15:0] at 0x0 2: Read B3B2[15:0] at 0x2 and write B3B2[15:0] at 0x2 3: Read B5B4[15:0] at 0x4, write B5B4[15:0] at 0x4 4: Read B7B6[15:0] at 0x6 and Write B7B6[15:0] at 0x6	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6
16	32	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: Read B1B0[7:0] at 0x0 and write 0000B1B0[31:0] at 0x0 2: Read B3B2[7:0] at 0x2 and write 0000B3B2[31:0] at 0x4 3: Read B5B4[7:0] at 0x4 and write 0000B5B4[31:0] at 0x8 4: Read B7B6[7:0] at 0x6 and write 0000B7B6[31:0] at 0xC	0x0/0000B1B0 0x4/000B3B2 0x8/0000B5B4 0xC/0000B7B6
32	8	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: Read B3B2B1B0 [31:0] at 0x0 and write B0 [7:0] at 0x0 2: Read B7B6B5B4 [31:0] at 0x4 and write B4 [7:0] at 0x1 3: Read BBBAB9B8 [31:0] at 0x8 and write B8 [7:0] at 0x2 4: Read BFBEBDBC [31:0] at 0xc and write BC [7:0] at 0x3	0x0/B0 0x1/B4 0x2/B8 0x3/BC
32	16	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: Read B3B2B1B0 [31:0] at 0x0 and write B1B0 [7:0] at 0x0 2: Read B7B6B5B4 [31:0] at 0x4 and write B5B4 [7:0] at 0x2 3: Read BBBAB9B8 [31:0] at 0x8 and write B9B8 [7:0] at 0x4 4: Read BFBEBDBC [31:0] at 0xc and write BDBC [7:0] at 0x6	0x0/B1B0 0x2/B5B4 0x4/B9B8 0x6/BDBC
32	32	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: Read B3B2B1B0 [31:0] at 0x0 and Write B3B2B1B0 [7:0] at 0x0 2: Read B7B6B5B4 [31:0] at 0x4 and write B7B6B5B4 [7:0] at 0x2 3: Read BBBAB9B8 [31:0] at 0x8 and write BBBAB9B8 [7:0] at 0x4 4: Read BFBEBDBC [31:0] at 0xc and write BFBEBDBC [7:0] at 0x6	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC

10.2.4.1. Operate an AHB device that does not support byte or half-word writes

If the DMA writes an AHB device that does not support byte or half-word write operations in bytes or half-words (i.e., HSIZE is not appropriate for this module), no error will occur and the DMA will write 32-bit HWDATA data as per the following two examples:

- When HSIZE=halfword, write halfword '0xABCD' and the DMA will set the HWDATA bus to '0xABCDABCD'.
- When HSIZE=byte, write byte '0xAB', DMA will set HWDATA bus to '0xABABABABAB'.

Assuming that the AHB/APB bridge is a 32-bit slave device to an AHB that does not handle the HSIZE parameter, it will transfer bytes or halfwords on any AHB to the APB in 32-bit as follows:

- A write byte data '0xB0' operation on an AHB to address 0x0 (or 0x1, 0x2, or 0x3) will be converted to a write word data '0xB0B0B0B0B0B0' operation on an APB to address 0x0 ' operation.
- A write halfword data '0xB1B0' operation on the AHB for address 0x0 (or 0x2) will be converted to a write data '0xB1B0B1B0' operation on the APB for address 0x0.

10.2.5. Error Handling

Reading or writing a reserved address area will generate a DMA transfer error. When a DMA transfer error occurs during a DMA read or write operation, the hardware automatically clears the EN bit in the Channel Configuration Register (DMA_CCRx) corresponding to the channel on which the error occurred, and that channel operation is halted. At this time, the Transmission Error Interrupt Flag Bit (TEIF) corresponding to that channel in the DMA_IFR register will be set, and an interrupt will be generated if the Transmission Error Interrupt Allow bit is set in the DMA_CCRx register.

The EN bit of the DMA_CCRx register cannot be set again by software (channel x reactivated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

When software receives a notification of a transmission error through a channel involving a peripheral, software first stops this peripheral in DMA mode in order to disable any waiting or future DMA requests. Software then typically reconfigures the DMA and peripheral in DMA mode for a new transfer.

10.2.6. Interrupt

Each DMA channel can generate interrupts on DMA transfer halves, transfer completions, and transfer errors. For application flexibility, these interrupts are turned on by setting different bits in the registers.

Table 10-2 DMA Interrupts

Interrupt Event	**Event Flag Bit**	**Enable Control Bit**
Halfway through transmission	HTIFx	HTIEx
Transmission complete	TCIFx	TCIEx
Transmission Error	TEIFx	TEIEx
Transmission halfway/transmission complete/transmission error	GIFx	-

Notes:

1) When the transmission length NDT is 1, the transmission half flag bit HTIF is not generated, and the TCIF bit is generated when the transmission is complete.

2) When the transmission length NDT is odd (greater than 1), both the HTIF and TCIF flags are generated. The internal signal TCIF will be generated at NDT=1; HTIF will be generated at (NDT-(NDT/2 (rounded) -1)). If NDT=5, TCIF will be generated when NDT is reduced to 1; HTIF will be generated when NDT is reduced to 4.

3) When the transmission length NDT is even (greater than 1), both the HTIF and TCIF flags are generated. The internal signal TCIF will be generated when NDT=1; HTIF will be generated when (NDT-(NDT/2 (rounded) -1)). If NDT=10, TCIF will be generated when NDT decreases to 1; HTIF will be generated when NDT decreases to 6.

10.2.7. DMA Peripheral Request Mapping

DMA peripheral requests are mapped to each channel of DMA1 (7 channels) or DMA2 (5 channels), controlled by the DMAx_MAP register of SYSCFG, and each peripheral request can be mapped to any of the 12 channels by configuration.

The relationship between the serial number and the peripheral request is shown in the table below:

Table 10-3 DMA Peripheral Request Mapping

Request for MUX input serial number	Source	Request for MUX input serial number	Source	Request for MUX input serial number	Source
0	ADC1	23	I2C2_RD	46	TIM5_CH1
1	ADC2	24	I2C2_WR	47	TIM5_CH2
2	ADC3	25	TIM1_CH1	48	TIM5_CH3
3	Reserved	26	TIM1_CH2	49	TIM5_CH4
4	Reserved	27	TIM1_CH3	50	TIM5_UP
5	SPI1_RD	28	TIM1_CH4	51	TIM5_TRIG
6	SPI1_WR	29	TIM1_COM	52	TIM6
7	SPI2_RD	30	TIM1_TRIG	53	TIM7
8	SPI2_WR	31	TIM1_UP	54	TIM8_CH1
9	SPI3_RD	32	TIM2_CH1	55	TIM8_CH2
10	SPI3_WR	33	TIM2_CH2	56	TIM8_CH3
11	USART1_RD	34	TIM2_CH3	57	TIM8_CH4
12	USART1_WR	35	TIM2_CH4	58	TIM8_COM
13	USART2_RD	36	TIM2_UP	59	TIM8_TRIG
14	USART2_WR	37	TIM3_CH1	60	TIM8_UP
15	USART3_RD	38	TIM3_CH3	61	TIM2_TRIG
16	USART3_WR	39	TIM3_CH4	62	TIM3_CH2
17	USART4_RD	40	TIM3_UP	63	TIM4_CH4
18	USART4_WR	41	TIM3_TRIG	64	TIM4_TRIG
19	USART5_RD	42	TIM4_CH1	65	ESMC_TX
20	USART5_WR	43	TIM4_CH2	66	ESMC_RX
21	I2C1_RD	44	TIM4_CH3	67	SDIO
22	I2C1_WR	45	TIM4_UP	68	USB

10.3. Register Descriptions (0x40020000)

10.3.1. DMA interrupt status register (DMA_ISR)

Address:0x00

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				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**
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:28	Reserved	-	-	Reserved
27	TEIF7	R	0	Channel 7 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no transmission error (TE); 1: Channel 7 transmission error (TE);
26	HTIF7	R	0	Channel 7 half-transmit flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transfer event; 1: a half-transfer event occurs on channel 7;
25	TCIF7	R	0	Channel 7 transmission completion flag. 0: no transmission completion (TC); 1: Channel 7 transmission complete (TC);
24	GIF7	R	0	Channel 7 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No TE/HT/TC event; 1: TE/HT/TC event occurred on channel 7;
23	TEIF6	R	0	Channel 6 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no transmission error (TE); 1: Channel 6 transmission error (TE);
22	HTIF6	R	0	Channel 6 half-transmit flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transfer event; 1: a half-transfer event occurs on channel 6;
21	TCIF6	R	0	Channel 6 transmission completion flag. 0: no transmission completion (TC);

Bit	Name	R/W	Reset Value	Function
				1: Channel 6 transmission complete (TC);
20	GIF6	R	0	Channel 6 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No TE/HT/TC event; 1: TE/HT/TC event occurred on channel 6;
19	TEIF5	R	0	Channel 5 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no transmission error (TE); 1: Channel 5 transmission error (TE);
18	HTIF5	R	0	Channel 5 half-transmit flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transfer event; 1: a half-transfer event occurs on channel 5;
17	TCIF5	R	0	Channel 5 transmission completion flag. 0: no transmission completion (TC); 1: Channel 5 transmission complete (TC);
16	GIF5	R	0	Channel 5 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No TE/HT/TC event; 1: TE/HT/TC event occurs on channel 5;
15	TEIF4	R	0	Channel 4 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no transmission error (TE); 1: Channel 4 transmission error (TE);
14	HTIF4	R	0	Channel 4 half-transmit flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transfer event; 1: a half-transfer event occurs on channel 4;
13	TCIF4	R	0	Channel 4 transmission completion flag. 0: no transmission completion (TC); 1: Channel 4 transmission complete (TC);
12	GIF4	R	0	Channel 4 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No TE/HT/TC event; 1: TE/HT/TC event occurs on channel 4;
11	TEIF3	R	0	Channel 3 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear.

Bit	Name	R/W	Reset Value	Function
				0: no transmission error (TE); 1: Channel 3 transmission error (TE);
10	HTIF3	R	0	Channel 3 half-transmit flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transfer event; 1: a half-transfer event occurs on channel 3;
9	TCIF3	R	0	Channel 3 transmission completion flag. 0: no transmission completion (TC); 1: Channel 3 transmission complete (TC);
8	GIF3	R	0	Channel 3 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No TE/HT/TC event; 1: TE/HT/TC event occurs on channel 3;
7	TEIF2	R	0	Channel 2 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no transmission error (TE); 1: Channel 2 transmission error (TE);
6	HTIF2	R	0	Channel 2 half-transmit flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transfer event; 1: a half-transfer event occurs on channel 2;
5	TCIF2	R	0	Channel 2 transmission completion flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no transmission completion (TC); 1: Channel 2 transmission complete (TC);
4	GIF2	R	0	Channel 2 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No TE/HT/TC event; 1: TE/HT/TC event occurred on channel 2;
3	TEIF1	R	0	Channel 1 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no transmission error (TE); 1: Channel 1 transmission error (TE);
2	HTIF1	R	0	Channel 1 half-transmit flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transfer event; 1: Channel 1 half-transfer event occurs;

Bit	Name	R/W	Reset Value	Function
1	TCIF1	R	0	Channel 1 transmission completion flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no transmission completion (TC); 1: Channel 1 transmission complete (TC);
0	GIF1	R	0	Channel 1 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No TE/HT/TC event; 1: TE/HT/TC event occurs on channel 1;

10.3.2. DMA Interrupt Flag Clear Register (DMA_IFCR)

Address:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				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
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31:28	Reserved	-	-	Reserved
27	CTEIF7	W	0	Channel 7 transmission error flag cleared. 0: No effect; 1: Clear TEIF7;
26	CHTIF7	W	0	Channel 7 half-transmission flag cleared. 0: No effect; 1: Clear HTIF7;
25	CTCIF7	W	0	Channel 7 transmission completion flag cleared. 0: No effect; 1: Clear TCIF7;
24	CGIF7	W	0	Channel 7 global interrupt flag cleared. 0: No effect; 1: Clear GIF/TEIF/HTIF/TCIF for channel 7;
23	CTEIF6	W	0	Channel 6 transmission error flag cleared. 0: No effect; 1: Clear TEIF6;
22	CHTIF6	W	0	Channel 6 half-transmission flag cleared. 0: No effect; 1: Clear HTIF6;
21	CTCIF6	W	0	Channel 6 transmission completion flag cleared.

Bit	Name	R/W	Reset Value	Function
				0: No effect; 1: Clear TCIF6;
20	CGIF6	W	0	Channel 6 global interrupt flag cleared. 0: No effect; 1: Clear GIF/TEIF/HTIF/TCIF for channel 6;
19	CTEIF5	W	0	Channel 5 transmission error flag cleared. 0: No effect; 1: Clear TEIF5;
18	CHTIF5	W	0	Channel 5 half-transmission flag cleared. 0: No effect; 1: Clear HTIF5;
17	CTCIF5	W	0	Channel 5 transmission completion flag cleared. 0: No effect; 1: Clear TCIF5;
16	CGIF5	W	0	Channel 5 global interrupt flag cleared. 0: No effect; 1: Clear GIF/TEIF/HTIF/TCIF for channel 5;
15	CTEIF4	W	0	Channel 4 transmission error flag cleared. 0: No effect; 1: Clear TEIF4;
14	CHTIF4	W	0	Channel 4 half-transmission flag cleared. 0: No effect; 1: Clear HTIF4;
13	CTCIF4	W	0	Channel 4 transmission completion flag cleared. 0: No effect; 1: Clear TCIF4;
12	CGIF4	W	0	Channel 4 global interrupt flag cleared. 0: No effect; 1: Clear GIF/TEIF/HTIF/TCIF for channel 4;
11	CTEIF3	W	0	Channel 3 transmission error flag cleared. 0: No effect; 1: Clear TEIF3;
10	CHTIF3	W	0	Channel 3 half-transmission flag cleared. 0: No effect; 1: Clear HTIF3;
9	CTCIF3	W	0	Channel 3 transmission completion flag cleared. 0: No effect; 1: Clear TCIF3;
8	CGIF3	W	0	Channel 3 global interrupt flag cleared. 0: No effect; 1: Clear GIF/TEIF/HTIF/TCIF for channel 3;
7	CTEIF2	W	0	Channel 2 transmission error flag cleared. 0: No effect; 1: Clear TEIF2;
6	CHTIF2	W	0	Channel 2 half-transmission flag cleared.

Bit	Name	R/W	Reset Value	Function
				0: No effect; 1: Clear HTIF2;
5	CTCIF2	W	0	Channel 2 transmission completion flag cleared. 0: No effect; 1: Clear TCIF2;
4	CGIF2	W	0	Channel 2 global interrupt flag cleared. 0: No effect; 1: Clear GIF/TEIF/HTIF/TCIF for channel 2;
3	CTEIF1	W	0	Channel 1 transmission error flag cleared. 0: No effect; 1: Clear TEIF1;
2	CHTIF1	W	0	Channel 1 half-transmission flag cleared. 0: No effect; 1: Clear HTIF1;
1	CTCIF1	W	0	Channel 1 transmission completion flag cleared. 0: No effect; 1: Clear TCIF1;
0	CGIF1	W	0	Channel 1 global interrupt flag cleared. 0: No effect; 1: Clear GIF/TEIF/HTIF/TCIF for channel 1;

10.3.3. DMA Channel 1 Configuration Register (DMA_CCR1)

Address:0x08

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:15	Reserved	-	-	Reserved
14	MEM2MEM	RW	0	Channel 1 memory to memory mode. 0: disable; 1: memory to memory mode enable;
13：12	PL[1:0]	RW	0	Channel 1 priority configuration. 00: low; 01: medium; 10: high; 11: very high;
11：10	MSIZE[1:0]	RW	0	Channel 1 memory data width.

Bit	Name	R/W	Reset Value	Function
				00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
9 : 8	PSIZE[1:0]	RW	0	Channel 1 peripheral data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
7	MINC	RW	0	Channel 1 memory address increment mode. 0: disabled; 1: memory address increment mode enable;
6	PINC	RW	0	Channel 1 peripheral address incremental mode. 0: disabled; 1: peripheral address increment mode enable;
5	CIRC	RW	0	Channel 1 cyclic mode. 0: disable; 1: cyclic mode enable;
4	DIR	RW	0	Channel 1 data transfer direction. 0: read from peripheral; 1: read from memory;
3	TEIE	RW	0	Channel 1 transmission error interrupt (TE) enable. 0: Disable; 1: TE interrupt enable;
2	HTIE	RW	0	Channel 1 half-transmission interrupt (HT) enable. 0: disabled; 1: HT interrupt enable;
1	TCIE	RW	0	Channel 1 transmission completion interrupt (TC) enable. 0: disabled; 1: TC interrupt enable;
0	EN	RW	0	Channel 1 enable. 0: Disable; 1: channel 1 enable;

10.3.4. DMA Channel 1 Number of Data Transfers Register (DMA_CNDTR1)

Address:0x0C

Reset value:0x0000 0000

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

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NDT[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-	-	Reserved
15：0	NDT[15:0]	RW	0	Channel 1 data transfer quantity. The number of data transfers is 0 to 65535.This register is written only when the channel is not active (DMA_CCR1.EN=0). This register is read-only after the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer. The contents of the register either become 0 when the data transfer is complete, or when the channel is configured for cyclic mode, the contents of the register are automatically reloaded with the value from the previous configuration. When the value of this register is 0, no data will be transferred even if the DMA channel starts.

10.3.5. DMA Channel 1 Peripheral Address Register (DMA_CPAR1)

Address:0x10

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	PA[31:0]	RW	0	Channel 1 Peripheral Address. The base address of the channel 1 peripheral data register, which is used as the source or destination of the data transfer. When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned to the halfword address. When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.6. DMA Channel 1 Memory Address Register (DMA_CMAR1)

Address:0x14

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	MA[31:0]	RW	0	Channel 1 memory address. Channel 1 memory address to be used as the source or destination of the data transfer. When MSIZE=2'b01, the MA[0] bit is not used. The operation is automatically aligned to the halfword address. When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.7. DMA Channel 2 Configuration Register (DMA_CCR2)

Address:0x1C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：15	Reserved	-	-	Reserved
14	MEM2MEM	RW	0	Channel 2 memory to memory mode. 0: disable; 1: memory to memory mode enable;
13：12	PL[1:0]	RW	0	Channel 2 Priority Configuration. 00: low;

Bit	Name	R/W	Reset Value	Function
				01: medium; 10: high; 11: very high;
11：10	MSIZE[1:0]	RW	0	Channel 2 memory data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
9：8	PSIZE[1:0]	RW	0	Channel 2 peripheral data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
7	MINC	RW	0	Channel 2 memory address increment mode. 0: disabled; 1: memory address increment mode enable;
6	PINC	RW	0	Channel 2 peripheral address incremental mode. 0: Disable; 1: peripheral address increment mode enable;
5	CIRC	RW	0	Channel 2 cyclic mode. 0: disable; 1: cyclic mode enable;
4	DIR	RW	0	Channel 2 data transfer direction. 0: read from peripheral; 1: read from memory;
3	TEIE	RW	0	Channel 2 transmission error interrupt (TE) enable. 0: Disable; 1: TE interrupt enable;
2	HTIE	RW	0	Channel 2 half-transmission interrupt (HT) enable. 0: disabled; 1: HT interrupt enable;
1	TCIE	RW	0	Channel 2 transmission completion interrupt (TC) enable. 0: disabled; 1: TC interrupt enable;
0	EN	RW	0	Channel 2 enable . 0: Disable; 1: channel 1 enable ;

10.3.8. DMA Channel 2 Number of Data Transfers Register (DMA_CNDTR2)

Address:0x20

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-	-	Reserved
15：0	NDT[15:0]	RW	0	Channel 2 data transfer quantity. The number of data transfers is 0 to 65535.This register is written only when the channel is not working (DMA_CCR2.EN=0). This register is read-only after the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer. The contents of the register either become 0 after the end of the data transfer, or when the channel is configured for cyclic mode, the contents of the register are automatically reloaded with the value from the previous configuration. When the value of this register is 0, no data will be transferred even if the DMA channel starts.

10.3.9. DMA Channel 2 Peripheral Address Register (DMA_CPAR2)

Address:0x24

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	PA[31:0]	RW	0	Channel 2 Peripheral Address. The base address of the channel 2 peripheral data register, which is used as the source or destination of the data transfer.

Bit	Name	R/W	Reset Value	Function
				When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned to the halfword address. When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.10. DMA Channel 2 Memory Address Register (DMA_CMAR2)

Address:0x28

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	MA[31:0]	RW	0	Channel 2 memory address. Channel 2 memory address to be used as the source or destination of the data transfer. When MSIZE=2'b01, the MA[0] bit is not used. The operation is automatically aligned to the halfword address. When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.11. DMA Channel 3 Configuration Register (DMA_CCR3)

Address:0x30

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：15	Reserved	-	-	Reserved
14	MEM2MEM	RW	0	Channel 3 memory to memory mode. 0: disable; 1: memory to memory mode enable;
13：12	PL[1:0]	RW	0	Channel 3 Priority Configuration. 00: low; 01: medium; 10: high; 11: very high;
11：10	MSIZE[1:0]	RW	0	Channel 3 memory data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
9：8	PSIZE[1:0]	RW	0	Channel 3 peripheral data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
7	MINC	RW	0	Channel 3 memory address increment mode. 0: disabled; 1: memory address increment mode enable;
6	PINC	RW	0	Channel 3 peripheral address incremental mode. 0: disabled; 1: peripheral address increment mode enable;
5	CIRC	RW	0	Channel 3 cyclic mode. 0: disable; 1: cyclic mode enable;
4	DIR	RW	0	Channel 3 data transfer direction. 0: read from peripheral; 1: read from memory;
3	TEIE	RW	0	Channel 3 transmission error interrupt (TE) enable. 0: Disable; 1: TE interrupt enable;
2	HTIE	RW	0	Channel 3 half-transmission interrupt (HT) enable. 0: disabled; 1: HT interrupt enable;
1	TCIE	RW	0	Channel 3 transmission completion interrupt (TC) enable. 0: disabled; 1: TC interrupt enable;
0	EN	RW	0	Channel 3 enable .

Bit	Name	R/W	Reset Value	Function
				0: Disable; 1: channel 1 enable ;

10.3.12. DMA Channel 3 Number of Data Transfers Register (DMA_CNDTR3)

Address:0x34

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-	-	Reserved
15：0	NDT[15:0]	RW	0	Channel 3 data transfer quantity. The number of data transfers is 0 to 65535.This register is written only when the channel is not working (DMA_CCR3.EN=0). This register is read-only after the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer. The contents of the register either become 0 when the data transfer is complete, or when the channel is configured for cyclic mode, the contents of the register are automatically reloaded with the value from the previous configuration. When the value of this register is 0, no data will be transferred even if the DMA channel starts.

10.3.13. DMA Channel 3 Peripheral Address Register (DMA_CPAR3)

Address:0x38

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	PA[31:0]	RW	0	Channel 3 Peripheral Address. The base address of the channel 3 peripheral data register, which is used as the source or destination of the data transfer. When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned with the halfword address. When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.14. DMA Channel 3 Memory Address Register (DMA_CMAR3)

Address:0x3C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	MA[31:0]	RW	0	Channel 3 memory address. Channel 3 memory address to be used as the source or destination of the data transfer. When MSIZE=2'b01, the MA[0] bit is not used. The operation is automatically aligned to the halfword address. When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.15. DMA Channel 4 Configuration Register (DMA_CCR4)

Address:0x44

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN

	RW	R W	R W	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R W

Bit	Name	R/W	Reset Value	Function
31 : 15	Reserved	-	-	Reserved
14	MEM2MEM	RW	0	Channel memory to memory mode. 0: disable; 1: memory to memory mode enable;
13 : 12	PL[1:0]	RW	0	Channel Priority Configuration. 00: low; 01: medium; 10: high; 11: very high;
11 : 10	MSIZE[1:0]	RW	0	Channel memory data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
9 : 8	PSIZE[1:0]	RW	0	Channel peripheral data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
7	MINC	RW	0	Channel memory address increment mode. 0: disable; 1: memory address increment mode enable;
6	PINC	RW	0	Channel peripheral address increment mode. 0: Disable; 1: peripheral address increment mode enable;
5	CIRC	RW	0	Channel cycling mode. 0: disable; 1: cyclic mode enable;
4	DIR	RW	0	Direction of channel data transfer. 0: read from peripheral; 1: read from memory;
3	TEIE	RW	0	Channel transmission error interrupt (TE) enable. 0: Disable; 1: TE interrupt enable;
2	HTIE	RW	0	Channel half-transfer interrupt (HT) enable. 0: Disable; 1: HT interrupt enable;
1	TCIE	RW	0	Channel transmission completion interrupt (TC) enable. 0: Disable;

Bit	Name	R/W	Reset Value	Function
				1: TC interrupt enable;
0	EN	RW	0	Channel Enable. 0: Disable; 1: channel enable;

10.3.16. DMA Channel 4 Number of Data Transfers Register (DMA_CNDTR4)

Address:0x48

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-	-	Reserved
15：0	NDT[15:0]	RW	0	Number of channel data transfers. The number of data transfers is 0 to 65535.This register is written only when the channel is not working (DMA_CCR3.EN=0). This register is read-only after the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer. The contents of the register either become 0 when the data transfer is complete, or when the channel is configured for cyclic mode, the contents of the register are automatically reloaded with the value from the previous configuration. When the value of this register is 0, no data will be transferred even if the DMA channel starts.

10.3.17. DMA Channel 4 Peripheral Address Register (DMA_CPAR4)

Address:0x4C

Reset value:0x0000 0000

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

PA[15:0]
RW

Bit	Name	R/W	Reset Value	Function
31：0	PA[31:0]	RW	0	Channel Peripheral Address. The base address of the channel peripheral data register, which is used as the source or destination of the data transfer. When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned to the halfword address. When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.18. DMA Channel 4 Memory Address Register (DMA_CMAR4)

Address:0x50

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	MA[31:0]	RW	0	Channel memory address. The channel memory address, which is used as the source or destination of the data transfer. When MSIZE=2'b01, the MA[0] bit is not used. The operation is automatically aligned to the halfword address. When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.19. DMA Channel 5 Configuration Register (DMA_CCR5)

Address:0x58

Reset value:0x0000 0000

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

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：15	Reserved	-	-	Reserved
14	MEM2MEM	RW	0	Channel memory to memory mode. 0: disable; 1: memory to memory mode enable;
13：12	PL[1:0]	RW	0	Channel Priority Configuration. 00: low; 01: medium; 10: high; 11: very high;
11：10	MSIZE[1:0]	RW	0	Channel memory data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
9：8	PSIZE[1:0]	RW	0	Channel peripheral data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
7	MINC	RW	0	Channel memory address increment mode. 0: disable; 1: memory address increment mode enable;
6	PINC	RW	0	Channel peripheral address increment mode. 0: Disable; 1: peripheral address increment mode enable;
5	CIRC	RW	0	Channel cycling mode. 0: disable; 1: cyclic mode enable;
4	DIR	RW	0	Direction of channel data transfer. 0: read from peripheral; 1: read from memory;
3	TEIE	RW	0	Channel transmission error interrupt (TE) enable. 0: Disable; 1: TE interrupt enable;
2	HTIE	RW	0	Channel half-transfer interrupt (HT) enable. 0: Disable; 1: HT interrupt enable;

Bit	Name	R/W	Reset Value	Function
1	TCIE	RW	0	Channel transmission completion interrupt (TC) enable. 0: Disable; 1: TC interrupt enable;
0	EN	RW	0	Channel Enable. 0: Disable; 1: channel enable;

10.3.20. DMA Channel 5 Number of Data Transfers Register (DMA_CNDTR5)

Address:0x5C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-	-	Reserved
15：0	NDT[15:0]	RW	0	Number of channel data transfers. The number of data transfers is 0 to 65535.This register is written only when the channel is not working (DMA_CCR3.EN=0). This register is read-only after the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer. The contents of the register either become 0 when the data transfer is complete, or when the channel is configured for cyclic mode, the contents of the register are automatically reloaded with the value from the previous configuration. When the value of this register is 0, no data will be transferred even if the DMA channel starts.

10.3.21. DMA Channel 5 Peripheral Address Register (DMA_CPAR5)

Address:0x60

Reset value:0x0000 0000

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

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

Bit	Name	R/W	Reset Value	Function
31：0	PA[31:0]	RW	0	Channel Peripheral Address. The base address of the channel peripheral data register, which is used as the source or destination of the data transfer. When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned to the halfword address. When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.22. DMA Channel 5 Memory Address Register (DMA_CMAR5)

Address:0x64

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	MA[31:0]	RW	0	Channel memory address. The channel memory address, which is used as the source or destination of the data transfer. When MSIZE=2'b01, the MA [0] bit is not used. The operation is automatically aligned to the halfword address. When MSIZE=2'b10, the MA [1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.23. DMA Channel 6 Configuration Register (DMA_CCR6)

Address:0x6C

Reset value:0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：15	Reserved	-	-	Reserved
14	MEM2MEM	RW	0	Channel memory to memory mode. 0: disable; 1: memory to memory mode enable;
13：12	PL[1:0]	RW	0	Channel Priority Configuration. 00: low; 01: medium; 10: high; 11: very high;
11：10	MSIZE[1:0]	RW	0	Channel memory data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
9：8	PSIZE[1:0]	RW	0	Channel peripheral data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
7	MINC	RW	0	Channel memory address increment mode. 0: disable; 1: memory address increment mode enable;
6	PINC	RW	0	Channel peripheral address increment mode. 0: Disable; 1: peripheral address increment mode enable;
5	CIRC	RW	0	Channel cycling mode. 0: disable; 1: cyclic mode enable;
4	DIR	RW	0	Direction of channel data transfer. 0: read from peripheral; 1: read from memory;
3	TEIE	RW	0	Channel transmission error interrupt (TE) enable. 0: Disable; 1: TE interrupt enable;
2	HTIE	RW	0	Channel half-transfer interrupt (HT) enable. 0: Disable;

Bit	Name	R/W	Reset Value	Function
				1: HT interrupt enable;
1	TCIE	RW	0	Channel transmission completion interrupt (TC) enable. 0: Disable; 1: TC interrupt enable;
0	EN	RW	0	Channel Enable. 0: Disable; 1: channel enable;

10.3.24. DMA Channel 6 Number of Data Transfers Register (DMA_CNDTR6)

Address:0x70

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-	-	Reserved
15：0	NDT[15:0]	RW	0	Number of channel data transfers. The number of data transfers is 0 to 65535.This register is written only when the channel is not working (DMA_CCR3.EN=0). This register is read-only after the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer. The contents of the register either become 0 when the data transfer is complete, or when the channel is configured for cyclic mode, the contents of the register are automatically reloaded with the value from the previous configuration. When the value of this register is 0, no data will be transferred even if the DMA channel starts.

10.3.25. DMA Channel 6 Peripheral Address Register (DMA_CPAR6)

Address:0x74

Reset value:0x0000 0000

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

PA[31:16]															
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**
PA[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：0	PA[31:0]	RW	0	Channel Peripheral Address. The base address of the channel peripheral data register, which is used as the source or destination of the data transfer. When PSIZE=2'b01, the PA [0] bit is not used. The operation is automatically aligned to the halfword address. When PSIZE=2'b10, the PA [1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.26. DMA Channel 6 Memory Address Register (DMA_CMAR6)

Address:0x78

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	MA[31:0]	RW	0	Channel memory address. The channel memory address, which is used as the source or destination of the data transfer. When MSIZE=2'b01, the MA [0] bit is not used. The operation is automatically aligned to the halfword address. When MSIZE=2'b10, the MA [1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.27. DMA Channel 7 Configuration Register (DMA_CCR7)

Address:0x80

Reset value:0x0000 0000

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

Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：15	Reserved	-	-	Reserved
14	MEM2MEM	RW	0	Channel memory to memory mode. 0: disable; 1: memory to memory mode enable;
13：12	PL[1:0]	RW	0	Channel Priority Configuration. 00: low; 01: medium; 10: high; 11: very high;
11：10	MSIZE[1:0]	RW	0	Channel memory data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
9：8	PSIZE[1:0]	RW	0	Channel peripheral data width. 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Reserved.
7	MINC	RW	0	Channel memory address increment mode. 0: disable; 1: memory address increment mode enable;
6	PINC	RW	0	Channel peripheral address increment mode. 0: Disable; 1: peripheral address increment mode enable;
5	CIRC	RW	0	Channel cycling mode. 0: disable; 1: cyclic mode enable;
4	DIR	RW	0	Direction of channel data transfer. 0: read from peripheral; 1: read from memory;
3	TEIE	RW	0	Channel transmission error interrupt (TE) enable. 0: Disable; 1: TE interrupt enable;
2	HTIE	RW	0	Channel half-transfer interrupt (HT) enable.

Bit	Name	R/W	Reset Value	Function
				0: Disable; 1: HT interrupt enable;
1	TCIE	RW	0	Channel transmission completion interrupt (TC) enable. 0: Disable; 1: TC interrupt enable;
0	EN	RW	0	Channel Enable. 0: Disable; 1: channel enable;

10.3.28. DMA Channel 7 Number of Data Transfers Register (DMA_CNDTR7)

Address:0x84

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：16	Reserved	-	-	Reserved
15：0	NDT[15:0]	RW	0	Number of channel data transmissions. The number of data transfers is 0 to 65535.This register is written only when the channel is not working (DMA_CCR3.EN=0). This register is read-only after the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer. The contents of the register either become 0 after the end of the data transfer, or when the channel is configured for cyclic mode, the contents of the register are automatically reloaded with the value from the previous configuration. When the value of this register is 0, no data will be transferred even if the DMA channel starts.

10.3.29. DMA Channel 7 Peripheral Address Register (DMA_CPAR7)

Address:0x88

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	PA[31:0]	RW	0	Channel Peripheral Address. The base address of the channel peripheral data register, which is used as the source or destination of the data transfer. When PSIZE=2'b01, the PA [0] bit is not used. The operation is automatically aligned to the halfword address. When PSIZE=2'b10, the PA [1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.30. DMA Channel 7 Memory Address Register (DMA_CMAR7)

Address:0x8C

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31：0	MA[31:0]	RW	0	Channel memory address. The channel memory address, which is used as the source or destination of the data transfer. When MSIZE=2'b01, the MA [0] bit is not used. The operation is automatically aligned to the halfword address. When MSIZE=2'b10, the MA [1:0] bits are not used. The operation is automatically aligned to the word address.

10.3.31. DMA Register Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4002 0000	DMA_ISR	Reserved				TEIF	HTIF	TCIF	GIF7	TEIF	HTIF	TCIF	GIF6	TEIF	HTIF	TCIF	GIF5	TEIF	HTIF	TCIF	GIF4	TEIF	HTIF	TCIF	GIF3	TEIF	HTIF	TCIF	GIF2	TEIF	HTIF	TCIF	GIF1
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DMA_IFCR	Reserved				CTEIF	CHTIF	CTCIF	CGIF7	CTEIF	CHTIF	CTCIF	CGIF6	CTEIF	CHTIF	CTCIF	CGIF5	CTEIF	CHTIF	CTCIF	CGIF4	CTEIF	CHTIF	CTCIF	CGIF3	CTEIF	CHTIF	CTCIF	CGIF2	CTEIF	CHTIF	CTCIF	CGIF1
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Reserved																	MEM2ME	PL[1:0]		MSIZE[1:1]		PSIZE[1:2]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	DMA_CNDTR1	Reserved																	NDT[15:0]														
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA_CPAR1	PA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	MA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved																																
0x01C	DMA_CCR2	Reserved																	MEM2ME	PL[1:0]		MSIZE[1:1]		PSIZE[1:2]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DMA_CNDTR2	Reserved																	NDT[15:0]														
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA_CPAR2	PA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CMAR2	MA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved																																
0x030	DMA_CCR3	Reserved																	MEM2ME	PL[1:0]		MSIZE[1:1]		PSIZE[1:2]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DMA_CNDTR3	Reserved																	NDT[15:0]														
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA_CPAR3	PA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_CMAR3	MA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	Reserved																																
0x044	DMA_CCR4	Reserved																	MEM2ME	PL[1:0]		MSIZE[1:1]		PSIZE[1:2]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	DMA_CNDTR4	Reserved																NDT[15:0]															
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	DMA_CPAR4	PA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	DMA_CMAR4	MA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved																																
0x058	DMA_CCR5	Reserved																	MEM2ME	PL[1:0]		MSIZE[1:1]		PSIZE[1:2]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	DMA_CNDTR5	Reserved																NDT[15:0]															
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DMA_CPAR5	PA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	DMA_CMAR5	MA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved																																
0x06C	DMA_CCR6	Reserved																	MEM2ME	PL[1:0]		MSIZE[1:1]		PSIZE[1:2]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	DMA_CNDTR6	Reserved																NDT[15:0]															
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	DMA_CPAR6	PA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	DMA_CMAR6	MA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved																																
0x080	DMA_CCR7	Reserved																	MEM2ME	PL[1:0]		MSIZE[1:1]		PSIZE[1:2]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x084	DMA_CNDTR7	Reserved																NDT[15:0]															
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	DMA_CPAR7	PA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	DMA_CMAR7	MA[31:0]																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11. Interrupts and events

11.1. Introduction

The Nested Vector Interrupt Controller (NVIC) interfaces closely with the processor core to enable low-latency interrupt handling and efficient handling of late arriving interrupts. The Nested Vector Interrupt Controller manages interrupts including kernel exceptions. Refer to the Cortex-M4 Programming Manual for more instructions on exceptions and NVIC programming.

11.1.1. Main features

- 60 maskable interrupt channels (16 Cortex™-M4 interrupt lines not included)
- 8 programmable priority levels (3-bit interrupt priority used)
- Low latency exception and interrupt handling
- Power management controls
- System control register implementation

11.1.2. Block diagrams of modules

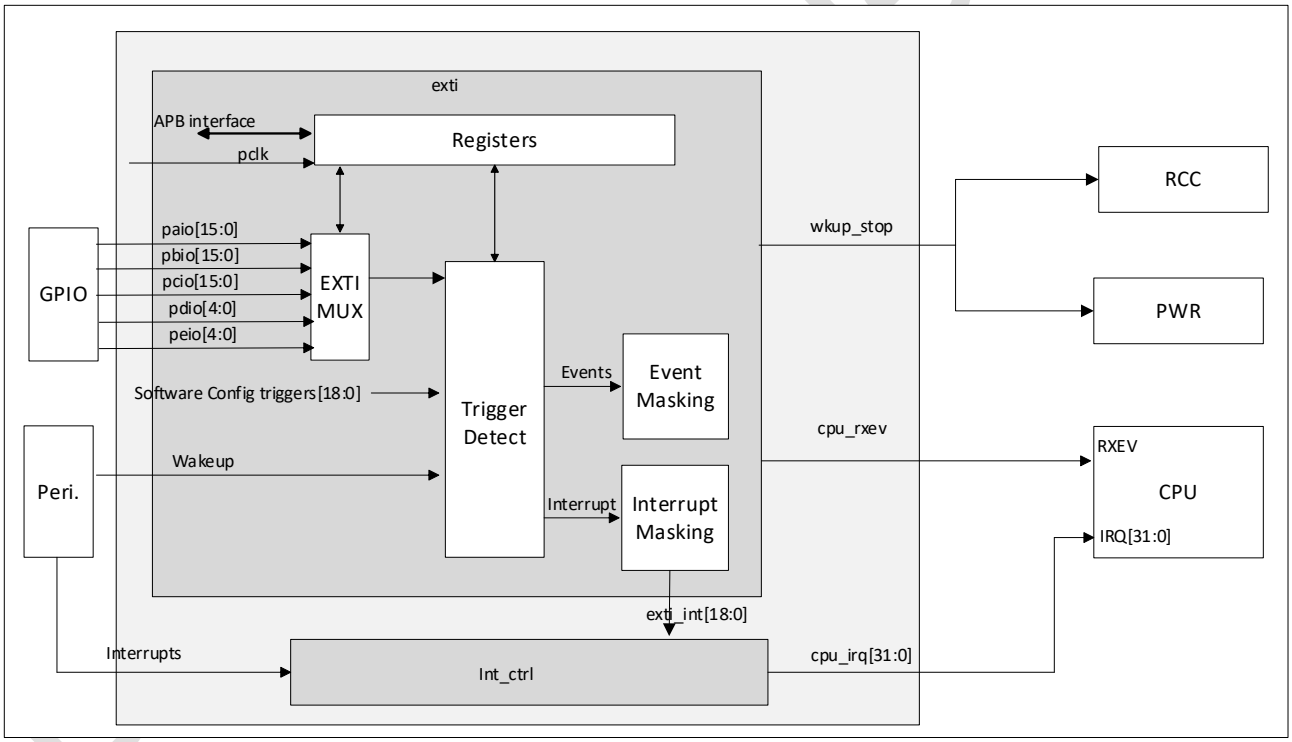


Figure 11-1 EXTI Module Block Diagram

11.2. Functional description

11.2.1. Interrupt and Exception Vector Table

Table 11-1 Interrupt and Exception Vector Table

Vector number	Priority	Priority type	Name	Address	Description
-	-	-	-	0x0000_0000	Reserved
-	-3	Fixed	Reset	0x0000_0004	Reset
-	-2	NMI	RCC HSE Clock Safety (HSE CSS)	0x0000_0008	unmaskable interrupt
-	-1	Fixed	hardware fault (Hard Fault)	0x0000_000C	All types of failure
-	0	Programmable	Mem Manage	0x0000_0010	memory management
	1	Programmable	Bus Fault	0x0000_0014	Prefetch failure, memory access failure
-	2	Programmable	Usage Fault	0x0000_0018	Undefined instructions or illegal states
-	-	-	-	0x0000_001C	Reserved
-	-	-	-	0x0000_0020	Reserved
-	-	-	-	0x0000_0024	Reserved
-	-	-	-	0x0000_0028	Reserved
-	3	Programmable	SVCall	0x0000_002C	System service calls via SWI instructions
-	4	Programmable	Debug Monitor	0x0000_0030	Debug Monitor
-	-	-	-	0x0000_0034	Reserved
-	5	Programmable	PendSV	0x0000_0038	Suspendable system services
-	6	Programmable	SysTick	0x0000_003C	System Tick Timer
0	7	Programmable	WWDG	0x0000_0040	Window Watchdog Timer Interrupt
1	8	Programmable	PVD	0x0000_0044	Supply voltage connected to EXTI is interrupted by detection (PVD)
2	9	Programmable	TAMPER	0x0000_0048	Intrusion detection interruption
3	10	Programmable	RTC	0x0000_004C	Real Time Clock (RTC) Global Interrupt
4	11	Programmable	FMC	0x0000_0050	Flash FMC Global Interrupt
5	12	Programmable	RCC 和 CTC	0x0000_0054	Reset and Clock Control (RCC) and CTC interrupts
6	13	Programmable	EXTI0	0x0000_0058	EXTI line 0 interrupts
7	14	Programmable	EXTI1	0x0000_005C	EXTI line 1 interrupts

Vector number	Priority	Priority type	Name	Address	Description
8	15	Programmable	EXTI2	0x0000_0060	EXTI line 2 interrupts
9	16	Programmable	EXTI3	0x0000_0064	EXTI line 3 interrupts
10	17	Programmable	EXTI4	0x0000_0068	EXTI line 4 interrupts
11	18	Programmable	DMA1 channel 1	0x0000_006C	DMA1 channel 1 globally interrupt
12	19	Programmable	DMA1 channel 2	0x0000_0070	DMA1 channel 2 globally interrupt
13	20	Programmable	DMA1 channel 3	0x0000_0074	DMA1 channel 3 globally interrupt
14	21	Programmable	DMA1 channel 4	0x0000_0078	DMA1 channel 4 globally interrupt
15	22	Programmable	DMA1 channel 5	0x0000_007C	DMA1 channel 5 globally interrupt
16	23	Programmable	DMA1 channel 6	0x0000_0080	DMA1 channel 6 globally interrupt
17	24	Programmable	DMA1 channel 7	0x0000_0084	DMA1 channel 7 globally interrupt
18	25	Programmable	ADC1_2	0x0000_0088	ADC1 and ADC2 Global Interrupt
19	26	Programmable	USB	0x0000_008C	USB interrupt
20	27	Programmable	CAN	0x0000_0090	CAN interrupt
21	28	Programmable	-	0x0000_0094	Reserved
22	29	Programmable	-	0x0000_0098	Reserved
23	30	Programmable	EXTI9_5	0x0000_009C	EXTI line [9:5] interruptions
24	31	Programmable	TIM1_BRK_TIM9	0x0000_00A0	TIMER1 abort interrupt and TIMER9 global interrupt
25	32	Programmable	TIM1_UP_TIM10	0x0000_00A4	TIMER1 abort interrupt and TIMER10 global interrupt
26	33	Programmable	TIM1_TRG_COM_TIM11	0x0000_00A8	TIMER1 abort interrupt and TIMER11 global interrupt
27	34	Programmable	TIM1_CC	0x0000_00AC	TIMER1 capture compare interrupt
28	35	Programmable	TIM2	0x0000_00B0	TIMER2 global interrupts

Vector number	Priority	Priority type	Name	Address	Description
29	36	Programmable	TIM3	0x0000_00B4	TIMER3 global interrupts
30	37	Programmable	TIM4	0x0000_00B8	TIMER4 global interrupts
31	38	Programmable	I2C1_EV	0x0000_00BC	I2C1event interrupt
32	39	Programmable	I2C1_ER	0x0000_00C0	I2C1 error interrupt
33	40	Programmable	I2C2_EV	0x0000_00C4	I2C2 event interrupt
34	41	Programmable	I2C2_ER	0x0000_00C8	I2C2 error interrupt
35	42	Programmable	SPI1	0x0000_00CC	SPI1 global interruptions
36	43	Programmable	SPI2	0x0000_00D0	SPI2 global interruptions
37	44	Programmable	USART1	0x0000_00D4	USART1 global interruptions
38	45	Programmable	USART2	0x0000_00D8	USART2 global interruptions
39	46	Programmable	USART3	0x0000_00DC	USART3 global interruptions
40	47	Programmable	EXTI15_10	0x0000_00E0	EXTI line [15:10] interruption
41	48	Programmable	RTCAlarm	0x0000_00E4	RTC Alarm Clock Interrupt Connected to EXTI
42	49	Programmable	-	0x0000_00E8	Reserved
43	50	Programmable	TIM8_BRK_TIM12	0x0000_00EC	TIMER8 abort interrupt and TIMER12 global interrupt
44	51	Programmable	TIM8_UP_TIM13	0x0000_00F0	TIMER8 Update Interrupt and TIMER13 Global Interrupt
45	52	Programmable	TIM8_TRG_COM_TIM14	0x0000_00F4	TIMER8 Trigger and Communication Interrupt TIMER14 Global Interrupt
46	53	Programmable	TIM8_CC	0x0000_00F8	TIMER8 Capture Compare Interrupt
47	54	Programmable	ADC3	0x0000_00FC	ADC3 global interrupt
48	55	Programmable	ESMC	0x0000_0100	ESMC global interrupt
49	56	Programmable	SDIO	0x0000_0104	SDIO global interrupt

Vector number	Priority	Priority type	Name	Address	Description
50	57	Programmable	TIM5	0x0000_0108	TIM5 global interrupt
51	58	Programmable	SPI3	0x0000_010C	SPI3 global interrupt
52	59	Programmable	USART4	0x0000_0110	USART4 global interrupt
53	60	Programmable	USART5	0x0000_0114	USART5 global interrupt
54	61	Programmable	TIM6	0x0000_0118	TIM6 global interrupt
55	62	Programmable	TIM7	0x0000_011C	TIM7 global interrupt
56	63	Programmable	DMA2 channel 1	0x0000_0120	DMA2 Channel 1 global interrupt
57	64	Programmable	DMA2 channel 2	0x0000_0124	DMA2 Channel 2 global interrupt
58	65	Programmable	DMA2 channel 3	0x0000_0128	DMA2 Channel 2 global interrupt
59	66	Programmable	DMA2 channel 4_5	0x0000_012C	DMA2 Channel 4 and Channe 5 global interrupt

11.2.2. External interrupt/event controller (EXIT)

There are 19 edge detectors capable of generating event/interrupt requests. Each input line can be independently configured with the input type (pulse or hang) and the corresponding trigger event (rising or falling edge or both edges triggered). Each input line can be independently masked. The pending register holds the interrupt request for the status line.

11.2.2.1. Main features

The main features of the EXTI controller are as follows:

- Each interrupt/event is independently triggered and masked
- Dedicated status bits for each interrupt line
- Supports up to 19 software requests for interrupts/events
- Detection of external signals with pulse widths lower than the APB2 clock width. See relevant parameters in the Electrical Characteristics section of the datasheet.

11.2.2.2. Wake-up event management

The P32F403xx can handle external or internal events to wake up the core (WFE). Wake-up events can be generated by the following configuration:

- Enable an interrupt in the control register of the peripheral but not in the NVIC, and enable the SEVONPEND bit in the system control register of the Cortex-M4. When the CPU recovers from

WFE, it is necessary to clear the interrupt pending bit of the corresponding peripheral and the peripheral NVIC interrupt channel pending bit (in the NVIC interrupt clear pending register).

- Configure an external or internal EXTI line to be in event mode, and when the CPU recovers from WFE, it is not necessary to clear the interrupt pending bit of the corresponding peripheral or the NVIC interrupt channel pending bit because the pending bit of the corresponding event line is not set.

To use an external I/O port as a wake-up event, see the following section.

11.2.2.3. Functional description

To generate an interrupt, the interrupt line must first be configured and enabled. Set the 2 trigger registers according to the desired edge detection and also write a '1' to the corresponding bit in the interrupt mask register to allow an interrupt request. When an expected edge occurs on the external interrupt line, an interrupt request is generated and the corresponding pending bit is set '1'. Writing a '1' to the corresponding bit in the pending register will clear the interrupt request. If an event needs to be generated, the event line must be configured and enabled first. Edge detection as required is allowed by setting the 2 trigger registers while writing a '1' to the corresponding bit in the event mask register to allow the event request. When the desired edge occurs on the event line, an event request pulse is generated and the corresponding pending bit is not set '1'. Interrupt/event requests can also be generated by software by writing a '1' to the software interrupt/event register.

Hardware Interrupt Selection

Configure the 19 lines as interrupt sources by following the procedure below:

- Configure the mask bits (EXTI_IMR) for the 19 interrupt lines
- Configure the trigger selection bits (EXTI_RTSR and EXTI_FTSR) for the selected interrupt lines;
- Configure the enable and mask bits of the NVIC interrupt channel corresponding to the external interrupt controller (EXTI), so that requests in the 19 interrupt lines can be responded to correctly.

Hardware Event Selection

- The following procedure allows you to configure 19 lines as event sources
- Configure the mask bits (EXTI_EMR) for the 19 event lines
- Configure the trigger selection bits (EXTI_RTSR and EXTI_FTSR) for the event lines

Software Interrupt/Event Selection

19 lines can be configured as software interrupt/event lines. The following is the procedure for generating a software interrupt:

- Configure the 19 interrupt/event line mask bits (EXTI_IMR, EXTI_EMR)
- Set the request bit of the software interrupt register (EXTI_SWIER)

11.2.2.4. External interrupt/event line map

The 80 general-purpose I/O ports are connected to 16 external interrupt/event lines in the manner shown below:

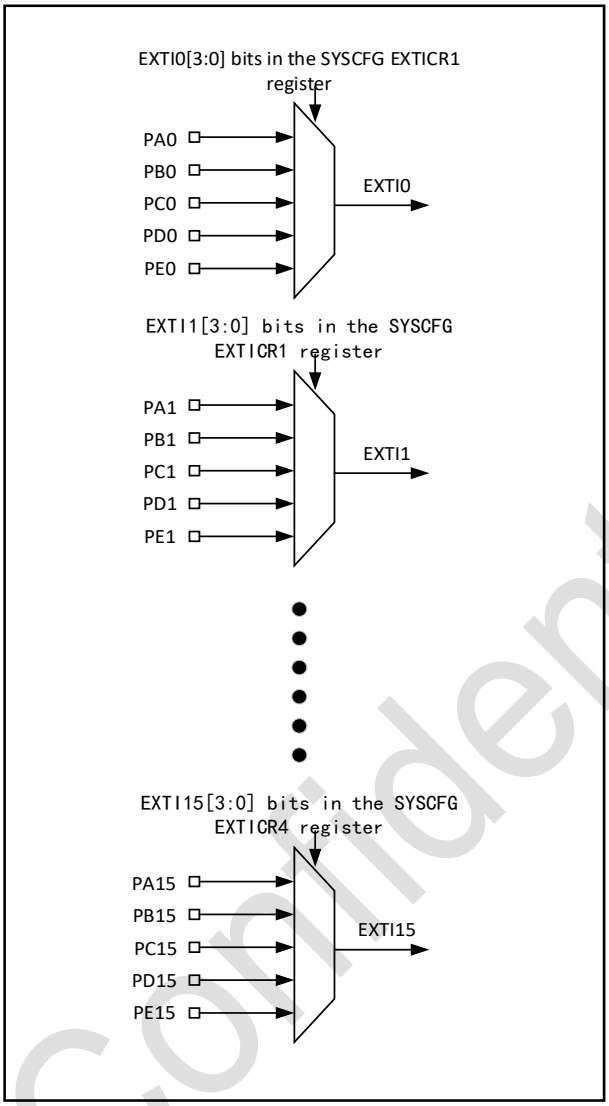


Figure 11-2 External Interrupt/Event Line Map

To configure external interrupts/events on the GPIO lines via SYSCFG_EXTICRx, the SYSCFG clock must be enabled first.

The other 2 EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD outputs
- EXTI line 17 is connected to the RTC alarm event

11.3. Register Descriptions

These registers must be accessed in a 32-bit manner.

11.3.1. Interrupt Mask Registers (EXTI_IMR)

Address offset:0x00

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	IMRx	RW	0	IMRx: Interrupt Mask on line x 0: Mask interrupt requests on line x; 1: Open interrupt request on line x.

11.3.2. Event Mask Register (EXTI_EMR)

Address offset:0x04

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													EMR 18	EMR 17	EMR 16
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**
EMR 15	EMR 14	EMR 13	EMR 12	EMR 11	EMR 10	EM R9	EM R8	EM R7	EM R6	EM R5	EM R4	EM R3	EMR 2	EMR 1	EMR 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	EMRx	RW	0	EMRx: Event Mask on line x 0: Mask event requests on line x; 1: open event requests on line x.

11.3.3. Rising-Edge Trigger Select Register (EXTI_RTST)

Address offset:0x08

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													RTS R18	RTS R17	RTS R16
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**
RTS R15	RTS R14	RTS R13	RTS R12	RTS R11	RTS R10	RT SR9	RT SR8	RT SR7	RT SR6	RT SR5	RT SR4	RT SR3	RTS R2	RTS R1	RTS R0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	RTSRx	RW	0	RTSRx: Rising trigger event con-figuration bit of line x 0: Rising trigger (interrupts and events) on line x is disabled. 1: Rising trigger event con-figuration bit of line x is allowed.

11.3.4. Falling-Edge Trigger Select Register (EXTI_FTSR)

Address offset:0x0C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													RTSR18	RTSR17	RTSR16
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**
FTSR15	FTSR14	FTFR13	FTSR12	FTSR11	FTSR10	FTSR9	FTSR8	FTSR7	FTSR6	FTSR5	FTSR4	FTSR3	FTSR2	FTSR1	FTSR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	FTSRx	RW	0	RTSRx: Falling trigger event configuration bit of line x 0: Falling trigger (interrupts and events) on line x is disabled 1: Falling trigger on line x allowed (interrupts and events)

11.3.5. Software Interrupt Event Register (EXTI_SWIER)

Address offset:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													SWIER18	SWIER17	SWIER16
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**
SWIER15	SWIER14	SWIER13	SWIER12	SWIER11	SWIER10	SWIER9	SWIER8	SWIER7	SWIER6	SWIER5	SWIER4	SWIER3	SWIER2	SWIER1	SWIER0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:19	Reserved			
18:0	SWIERx	RW	0	SWIERx: Software interrupt on line x. When this bit is '0', writing '1' will set the corresponding pending bit in EXTI_PR. If this interrupt is allowed to be generated in EXTI_IMR and EXTI_EMR, it will be generated at this time. to generate this interrupt, then an interrupt will be generated at this time. Note: This bit can be cleared to '0' by clearing the corresponding bit in EXTI_PR (writing '1').

11.3.6. Suspend register (EXTI_PR)

Address offset:0x14

Reset value:0x000X XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													PR18	PR17	PR16
													rc_w1	rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
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

Bit	Name	R/W	Reset Value	Function
31: 19	Reserved			
18: 0	PRx	RC_W1	x	PRx: Pending bit 0: No trigger request occurred 1: A selected trigger request has occurred This bit is set to '1' when a selected edge event occurs on the external interrupt line. Writing a '1' to this bit clears it, or it can be cleared by changing the polarity of the edge detection.

11.3.7. EXTIregister address map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_IMR	Reserved													MR[18:0]																		
	Read/Write														rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	EXTI_EMR	Reserved													MR[18:0]																		
	Read/Write														rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	EXTI_RTSR	Reserved													TR[18:0]																		
	Read/Write														rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset Value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	EXTI_FTSR	Reserved													TR[18:0]																			
	Read/Write														rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	EXTI_SWIER	Reserved													SWIER[18:0]																			
	Read/Write														rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	EXTI_PR	Reserved													PR[18:0]																			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read/Write														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	rc_w1	rc_w1	rc_w1	rc_w1
	Reset Value														x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

12. Analogue/digital conversion (ADC)

12.1. Introduction

A 12-bit ADC is an analogue-to-digital converter that uses successive approximation. It has 20 multiplexed channels and can convert analogue signals from 16 external channels and 3 internal channels. An analogue watchdog allows applications to detect if the input voltage exceeds a user-set high or low threshold. The A/D conversion of the various channels can be configured in single, continuous, sweep or intermittent conversion modes. The results of the ADC conversions can be stored in 16-bit data registers in a left- or right-aligned manner.

12.1.1. Main features

- High Performance.
  - Configurable 12 bits, 10 bits, 8 bits and 6 bits resolutions;
  - Maximum ADC sampling rate: 1MSPs.
  - Self-calibrating;
  - Programmable sample time;
  - Data register configurable data alignment;
  - DMA request support for regular channel data conversion.
  - Dual ADC mode (with 2 or more ADC devices)
- Analogue input channels.
  - 16 channels of external analog inputs;
  - 1 internal temperature sensing channel (VSENSE);
  - 1 internal reference voltage input channel (VREFINT).
  - battery detection input channel (VBAT)
- Initiate conversion method:
  - Software start-up
  - Hardware Trigger
- conversion mode:
  - Single mode, which converts the selected input channel once per trigger;
  - Scan mode, which scans a series of channels;
  - Continuous mode, which continuously converts the selected input channels;
  - Intermittent mode, which continuously converts a subsequence of channels per trigger, and multiple triggers until all channels are converted.
  - synchronised mode (for devices with two or more ADCs).
- Analogue watchdog
- Interrupt generation:
  - End of rule group or injection group conversion
  - Analogue watchdog event

- ADC supply requirements: 1.7V to 3.6V, typical supply voltage is 3.3V.
- ADC input range: VREF- ≤ VIN ≤ VREF+

12.1.2. Module block diagram

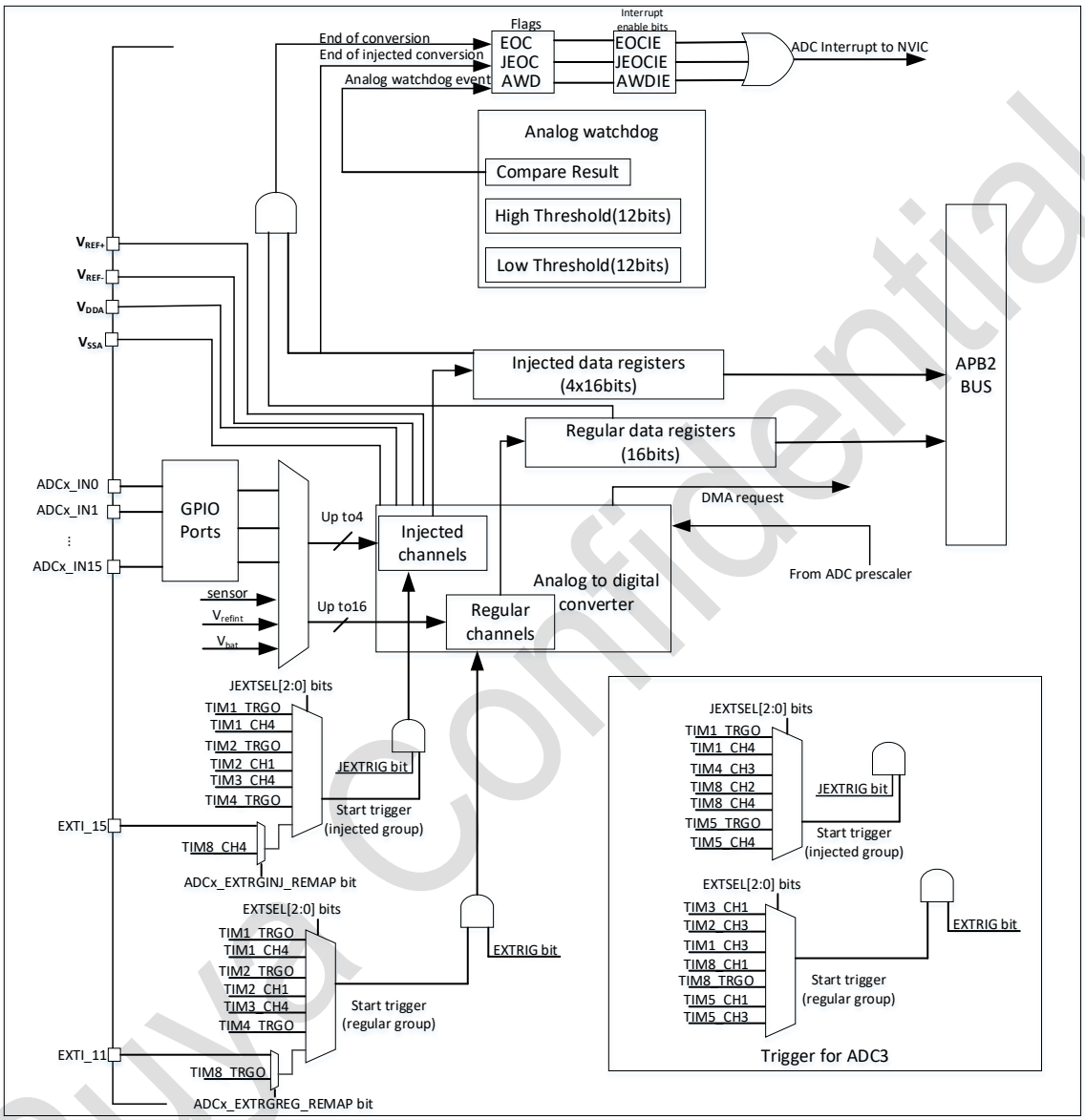


Figure 12-1 ADC module block diagram

12.2. ADC Pin Definitions

Table 12-1 ADC Pin Definitions

Name	Signal type	Remark
VDDA	Input, analogue power supply	VDD-equivalent analogue power supply. 2.0V ≤ VDDA ≤ VDD for full speed operation 1.7V ≤ VDDA ≤ VDD for low speed operation
VSSA	Input, analogue power ground	Analogue power ground equivalent to VSS

Name	Signal type	Remark
V_{REF+}	Input, analogue reference positive	Positive reference voltage used by the ADC, 1.7V ≤ VREF+ ≤ VDDA
V_{REF-}	Input, analogue reference negative	Negative reference voltage used by the ADC, VREF- = VSSA
ADCx_IN[15:0]	Input, analogue signal	Analogue input channels

12.3. Functional descriptions

12.3.1. ADC calibration

The ADC has a calibration function that can be enabled by the user through software. During calibration, the ADC calculates a calibration factor that is used internally by the ADC (lost when the ADC loses power). During ADC calibration, the application cannot use the ADC module until calibration is complete. It is recommended that the user perform a calibration operation before using ADC conversion. Calibration is used to eliminate chip-to-chip misalignment and mismatch errors due to process variations. The calibration operation is a software calibration.

12.3.1.1. ADC Software Calibration

A software setting of CAL=1 initiates calibration, which can only be initiated when the ADC is not turned on (ADON=0) and only supports selection of the system clock as the ADC clock. When calibration is complete, CAL is cleared by hardware to 0. The calibration factor is maintained until a system reset is generated. When the operating conditions of the ADC change (VCC change is the main factor of the misalignment error and mismatch error, followed by temperature change), it is recommended to perform another calibration operation. Software procedure for calibration:

- Confirm ADON=0
- Set CAL=1
- Wait until CAL=0

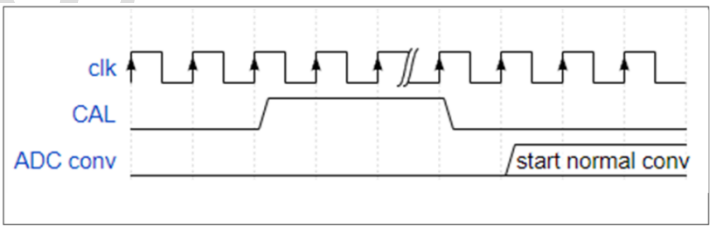


Figure 12-2ADC Calibration Timing Chart

12.3.2. ADC on-off control

The ADON bit in the ADC_CR1 register is the enable switch for the ADC module. To save power, the ADC analogue sub-module will enter power-down mode when ADON is 0.

Caution:

1) After the ADON bit is set to 1, you need to wait for not less than 20us delay (tSTAB) before conversion.

2) Conversion can be stopped by clearing the ADON bit and placing the ADC analogue sub-module in power-down mode. the ADC requires a stabilisation time of tSTAB before it can start an accurate conversion. after the ADC channel group has completed the conversion, the EOC flag is set and the 16-bit ADC data register contains the result of the conversion.

12.3.3. ADC Clock

The ADCCLK clock provided by the clock controller is synchronised with PCLK2 (APB2 clock). the RCC controller (CLK controller) provides a dedicated programmable prescaler for the ADC clock.

12.3.4. ADC Channel Selection

The ADC has 16 external channels and 3 internal channels, of which the internal channels are: temperature sensor VSENSE, VREFINT and VBAT. 16 external channels are connected to ADC123_IN0-ADC123_IN15, and of the 4 internal channels, the temperature sensor VSENSE is internally connected to channel ADC1_IN16, the internal reference voltage VREFINT is connected to ADC1_IN17, and the BAT voltage VBAT is connected to ADC123_IN18. VREFINT is connected to ADC1_IN17, BAT voltage VBAT is connected to ADC123_IN18.

ADC channel selection can be configured through ADC_SQRx or ADC_JSQR, and the configuration values and channel correspondences are shown in the following table:

Table 12-2 Channel relationship table

Configuration values	channel
00000	ADC analogue input channel 0
00001	ADC analogue input channel 1
00010	ADC analogue input channel 2
…	…
01111	ADC analogue input channel 15
10000	ADC analogue input channel 16
10001	ADC analogue input channel 17
10010	ADC analogue input channel 18
10011	ADC analogue input channel 19

The ADC divides the conversions into two groups: regular conversions and injected conversions. Each group contains a sequence of conversions that can be done on any channel in any order. For example, the sequence can be converted in the following order: ADC_IN3, ADC_IN8, ADC_IN2, ADC_IN2, ADC_IN0, ADC_IN2, ADC_IN2, ADC_IN15 All channels can be converted in either injected or regular channel groups.

Note: Temperature sensor, VREFINT channels are only available on the ADC1 peripheral.

- A rule conversion group consists of up to 16 conversions. The rule channel of the conversion sequence and its conversion order are selected in the ADC_SQRx register. The length of the sequence in the rule conversion group must be written to the L[3:0] bits in the ADC_SQR1 register.
- An injection conversion group consists of up to 4 conversions. The injection channel and its conversion order are selected in the ADC_JSQR register. The length of the sequence in the injected conversion group must be written to bits L[1:0] in the ADC_JSQR register.

If the ADC_SQRx or ADC_JSQR registers are modified during the conversion of each channel group, the current conversion is cleared and a new start pulse is sent to the ADC to convert the newly selected group.

12.3.5. Force stopping ADC

Setting ADSTP=1 in the ADC_CR1 register by software stops the current conversion in progress and puts the ADC into an idle state ready for the next conversion.

When ADSTP is set to 1 by software, any current conversion is aborted and the result is discarded (ADC_DR register is not updated with the current conversion value).

Once the conversion process is finished, ADSTP is cleared to 0 by hardware.

12.3.6. Conversion mode

(Note: When ADCON=1, ADC can change the conversion mode of ADC during the conversion process, but the result of ADC conversion will be abnormal, so it is not recommended to change the conversion mode during the ADC conversion process.)

12.3.6.1. Single conversion mode

In single-conversion mode, the ADC performs a single conversion for a single channel, and this mode is capable of operating in both rule group and injection group. Bits SQ1[4:0] of the ADC_SQR3 register specifies the ADC rule group single-conversion channel; for the injection channel single-conversion, the application should keep ADC_JSQR [21:20]: JL to 0, and bits JSQR4 of the ADC_JSQR register [4:0] bits configure the injection group single conversion channel.

When the ADON bit is 1 and CONT, SCAN and DISCEN/JDISCEN are all 0, the ADC samples and converts one channel once the corresponding external trigger occurs. (Note: Single mode and continuous mode cannot be enabled at the same time, i.e. CONT is 0 in single mode.)

- If a rule channel is converted:
  - the conversion data is stored in the 16-bit ADC_DR register
  - The EOC (end of conversion) flag is set after the conversion is complete
  - If EOCIE is set, an interrupt is generated.
- If an injection channel is converted:
  - The conversion data is stored in the 16-bit ADC_JDRx register
  - The JEOC (end of injection conversion) flag is set after the conversion is completed
  - If the JEOCIE bit is set, an interrupt is generated.

12.3.6.2. Continuous Conversion Mode

In continuous conversion mode, the ADC performs continuous conversion for the selected channel, and this mode is able to run in rule group.

- Rule channel group: the channel for continuous conversion is set by the SQ1[4:0] bits of the ADC_SQR3 register.
- Injection channel group: only a single conversion is executed, and the converted channel is set by the SQ1[4:0] bits of the ADC_SQR3 register.

When the ADON bit is 1 and CONT is 1, the ADC samples and converts the selected channel once the corresponding external trigger occurs.

- If a rule channel is converted:
- The conversion data is stored in the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set after the conversion is complete
- If EOCIE is set, an interrupt is generated.
- If an injection channel is converted:
- The conversion data is stored in the 16-bit ADC_JDR1 register
- The JEOC (end of injection conversion) flag is set after the conversion is completed
- If the JEOCIE bit is set, an interrupt is generated.

12.3.6.3. Scan Mode

In scan conversion mode, the ADC converts all channels of a selected sequence consecutively, and this mode is capable of operating in both rule group and injection group. The ADC_SQRx specifies a rule sequence length as well as all converted channels, where the L [3:0] bits of the ADC_SQR1 register specify the sequence length. The ADC_JSQR register specifies an injection group sequence length as well as all conversion channels, where the JL [1:0] bits of the ADC_JSQR register specify that sequence length.

When the ADON bit is 1 and SCAN is 1, the ADC samples and converts a sequence once the corresponding external trigger occurs.

- If a regular sequence is converted:
- The conversion data is stored in the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set after the conversion is complete
- If EOCIE is set, an interrupt is generated.
- If an injection sequence is converted:
- The conversion data is stored in the 16-bit ADC_JDRx register
- The JEOC (end of injection conversion) flag is set after the conversion is completed
- If the JEOCIE bit is set, an interrupt is generated.

12.3.6.4. Intermittent conversion mode

In intermittent conversion mode, the ADC divides a selected sequence into sub-sequences (sequence lengths 1-8), and completes the entire sequence conversion by externally triggering the sub-sequence multiple times, which is capable of operating in both rule group and injection group. The DISCNUM of ADC_CR1 specifies the length of the sub-sequence of the rule channel. The ADC_SQRx specifies all the conversion channels of a rule sequence where the L [3:0] bits of the ADC_ L [3:0] bits of the SQR1 register specify the sequence length. The ADC_JSQR register specifies an injection group sequence all conversion channels, where the JL [1:0] bits of the ADC_JSQR register specify the length of the sequence.

Note: The subsequence of the injected channel group is 1. This mode disables both discen and jdiscen from being enabled at the same time.

- Rule Group

An external trigger signal starts ADC_SQRx describing channel n (n<=8) conversions until all conversions of this sequence are completed. The total sequence length is defined by L[3:0] in the ADC_SQR1 register.

Example:

n=3, channels to be converted = 0, 1, 2, 3, 6, 7, 9, 10

First trigger: the converted sequence is 0, 1, 2

Second trigger: the converted sequence is 3, 6, 7

Third trigger: the converted sequence is 9, 10 and an EOC event is generated

Fourth trigger: converted sequence 0, 1, 2

Note: When a rule group is converted in interrupted mode, the converted sequence does not automatically start from the beginning when it is finished. When all subgroups have been converted, the next trigger starts the conversion of the first subgroup. In the example above, the fourth trigger reconverts channels 0, 1 and 2 of the first subgroup.

- Injection Group

An external trigger signal initiates ADC_JSQR to describe channel 1 conversions until all conversions in the sequence have been completed. The total sequence length is defined by the JL[1:0] bits of the ADC_JSQR register.

Example:

1) n=1, channel being converted = 1, 2, 3

First trigger: channel 1 is converted

Second trigger: channel 2 is converted

Third trigger: channel 3 is converted and EOC and JEOC events are generated.

Fourth trigger: channel 1 is converted

Note:

1) When all injection channels are converted, the next trigger initiates the conversion of the first injection channel. In the above example, the fourth trigger re-converts the first injection channel 1.

2) Automatic injection and intermittent mode cannot be used at the same time.

3) You must avoid setting interrupt mode for rules and injection groups at the same time. Intermittent mode can only be used for one set of transitions.

12.3.7. Injection channel management

The injection channel external trigger has a higher priority than the rule channel external trigger, i.e. the injection channel external trigger can interrupt the ongoing rule channel conversion. There are two ways to interrupt the injection channel: triggered injection and automatic injection.

12.3.7.1. Trigger injection

When the ADC detects an external trigger on the injection channel during the execution of a rule sequence, it interrupts the channel on which the current rule sequence is being converted and starts to execute the injection sequence conversion, and when all the conversions of the injection sequence are finished, the ADC restores the interrupted rule channel and completes the rule sequence conversion.

ADC_SQRx specifies all conversion channels of a rule sequence, where the L[3:0] bits of the ADC_SQR1 register specify the length of the sequence. The ADC_JSQR register specifies all conversion channels of an injection group sequence, where the JL[1:0] bits of the ADC_JSQR register specify the length of the sequence.

Note: When triggering an injection, JAUTO must be 0. If a rule-triggered event occurs during the conversion of an injection channel when in scan mode, the event converts the injection channel after it is completed and the original interrupted rule channel is discarded.

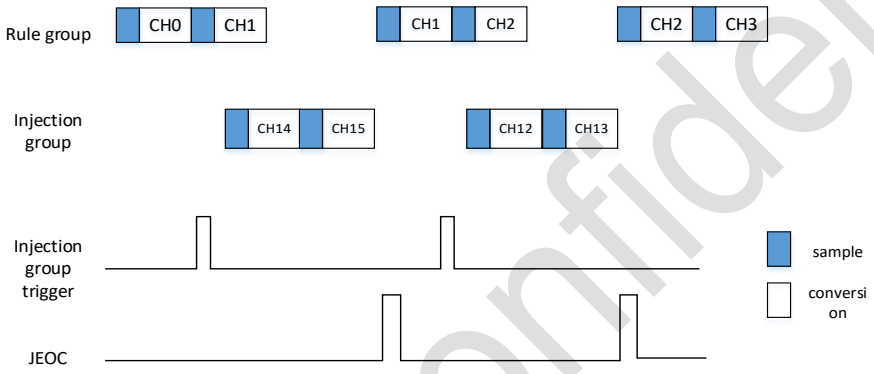


Figure 12-3 ADC trigger injection (scan mode)

12.3.7.2. Automatic Injection

In auto-injection mode, the ADC automatically performs injection sequence conversions after completing a rule sequence.JAUTO of ADC_CR1 is the auto-injection mode enable bit.ADC_SQR1, ADC_SQR2, and ADC_SQR3 specify all conversion channels of a rule sequence, where the L[3:0] bits of the ADC_SQR1 register specify the length of the sequence. The ADC_JSQR registers specify an injection group sequence of all conversion channels, where the JL[1:0] bits of the ADC_JSQR register specify the length of the sequence.

(Note that external triggering of the injection channels must be disabled in this mode; auto-inject and interrupt modes cannot be used simultaneously)

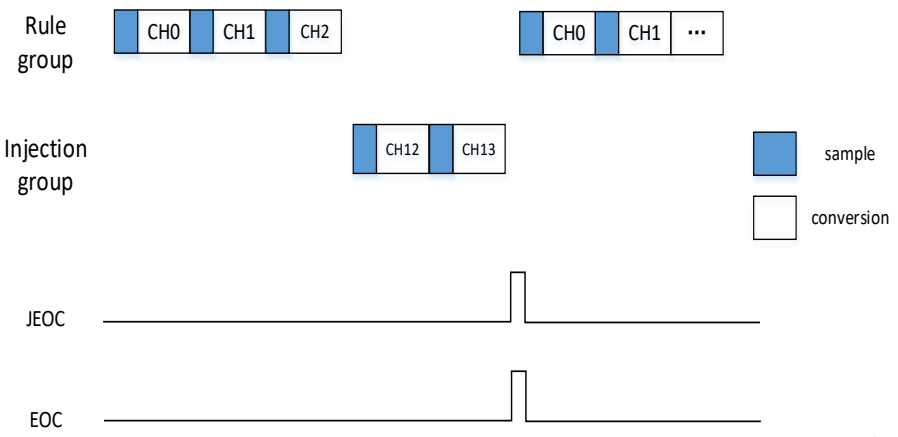


Figure 12-4 ADC auto-injection (cont+scan+jauto)

12.3.8. Analogue Watchdog

The AWD analogue watchdog status bits are set to 1 if the analogue voltage converted by the ADC is below the lower threshold or above the upper threshold. These thresholds are set in the 12 least significant bits of the ADC_HTR and ADC_LTR registers. An interrupt is generated by setting the AWDIE bit in the ADC_CR1 register.

The thresholds are independent of the selected alignment of the ALIGN bit in the ADC_CR2 register. Threshold comparisons are done prior to the alignment (before injecting the channel minus the offset value).

By configuring the ADC_CR1 register, the analogue watchdog can act on 1 or more channels as shown in the table below:

Table 12-3 Simulated watchdog channel selection

Analogue watchdog protection channel	ADC_CR1 register control bits		
	AWDSGL	AWDEN	JAWDEN
None	X	0	0
All Injection Channels	0	0	1
All rule channels	0	1	0
All Injection and Rule Channels	0	1	1
Single Injection Channel	1	0	1
Single Rule Channel	1	1	0
Single injection or rule channel	1	1	1

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

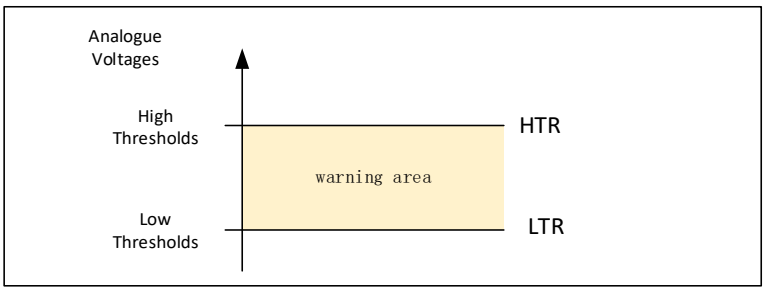


Figure 12-5 Analogue watchdog protection area

12.3.9. Data Alignment

The ALIGN bit in the ADC_CR2 register is used to select the alignment of the data stored after conversion, and can be selected as either left-aligned or right-aligned. The data value injected into the channel group conversion has been subtracted from the custom offset in the ADC_JOFRx register, so the result can be a negative value. The SEXT bit indicates the extended sign value.

For channels in a 12bit rule group, no offset is subtracted, so only 12 bits are valid.

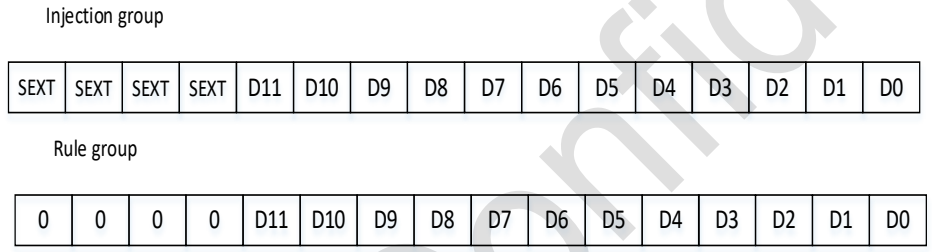


Figure 12-6 12-bit data right-aligned

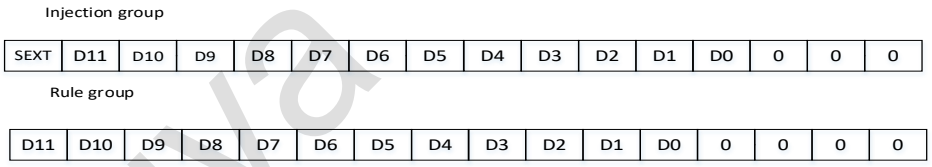


Figure 12-7 12-bit data left-aligned

12.3.10. Programmable sampling time

The ADC samples the input voltage over a number of ADC CLK cycles, the number of which is configured by the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. Each channel can be sampled using a different sampling time.

Note: The ADC CH18:VBAT channel sample times share a sample time configuration register (SMP0[2:0]) with ADC CH0.

The total conversion time is calculated as follows:

Tconv = Sample Time + 12.5 cycles

Example:

ADCCLK = 16MHz and Sample Time = 3.5 cycles:

Tconv = 3.5 + 12.5 = 16 cycles = 1 μs

12.3.11. Externally triggered conversion

Conversions can be triggered by external events (e.g., timer capture, EXTI interrupt). If the EXTTRIG or JEXTTRIG control bits are set, an external event is able to trigger a conversion. the EXTSEL[2:0] and JEXTSEL2:0] control bits allow the application to select one of eight possible events that can trigger the sampling of the rule and injection groups.

When an external trigger signal is selected for an ADC rule or injection conversion, only the rising edge can initiate the conversion.

Table 12-4 External Trigger for ADC1,ADC2 Rule Channel

Trigger source	Type	EXTSEL[2:0]
CH1 output of timer 1	Internal signals for on-chip timer	000
CH2 output of timer 1		001
CH3 output of timer 1		010
CH2 output of timer 2		011
TRGO output of timer 3		100
CH4 output of timer 4		101
EXTI line 11 or TRGO output of timer 8 (1)	External pin or internal signal from Timer 8	110
SWSTART	software control bits	111

(1). Select ADC1_ETRGREG_REMAP, ADC2_ETRGREG_REMAP by setting the SYSCFG_CFGR2 register in the SYSCFG module's

Table 12-5 External Trigger for ADC1,ADC2 Injection Channels

Trigger Source	Type	JEXTSEL[2:0]
TRGO output of Timer 1	Internal signals for on-chip timer	000
CH4 output of Timer1		001
TRGO output of Timer 2		010
CH1 output of Timer 2		011
CH4 output of Timer 3		100
TRGO output of Timer 4		101
EXTI line 15 / CH4 output of timer 8 (2)	External pins or internal signals for Timer 8	110
JSWSTART	Software control bits	111

(2). Select ADC1_ETRGINJ_REMAP, ADC2_ETRGINJ_REMAP by setting the SYSCFG_CFGR2 register in the SYSCFG module's

Table 12-6 External Trigger for ADC3 Rule Channel

Trigger Source	Type	EXTSEL[2:0]
CH1 output of timer 3	Internal signals for on-chip timer	000
CH3 output of timer 2		001
CH3 output of Timer 1		010
CH1 output of Timer 8		011
TRGO output of timer 8		100
CH1 output of Timer 5		101
CH3 output of Timer 5		110
SWSTART	Software Control Bits	111

Table 12-7External triggers for ADC3 injection channels

Trigger Source	Type	JEXTSEL[2:0]
TRGO output of Timer 1	Internal signals for on-chip timer	000
CH4 output of timer 1		001
CH3 output of timer 4		010
CH2 output of Timer 8		011
CH4 output of Timer 8		100
TRGO output of Timer 5		101
CH4 output of Timer 5		110
JSWSTART	Software Control Bits	111

12.3.12. Configurable resolution

Fast conversions can be performed by reducing the ADC resolution. The RESSEL bit in the ADC_CR1 register is used to select the number of bits available in the data register. The minimum conversion times for each resolution are as follows:

- 12-bit: 3.5 + 12.5 = 16 ADCCLK cycles
- 10-bit: 3.5 + 10.5 = 14 ADCCLK cycles
- 8-bit: 3.5 + 8.5 = 12 ADCCLK cycles
- 6-bit: 3.5 + 6.5 = 10 ADCCLK cycles

12.3.13. DMA request

Because the values of rule channel conversions are stored in a unique data register, DMA is required when converting multiple rule channels, which prevents the loss of data already stored in the ADC_DR register.

A DMA request is generated only when the conversion of a rule channel is complete and the converted data is transferred from the ADC_DR register to the user-specified destination address. In dual DMA mode, ADC2 converts the data and transfers it using the DMA function of ADC1.

12.3.14. Dual-ADC mode

In devices with 2 ADCs, dual ADC mode can be used. In Dual ADC mode, the conversion can be initiated by alternating or simultaneous triggering of ADC1 master and ADC2 slave, depending on the mode selected by the DUALMOD [2:0] bits in the ADC1_CR1 register.

Note: In Dual ADC mode, when the conversion is configured to be triggered by an external event, the user must set it to trigger the master ADC only, and set the slave ADC to be triggered by software, so as to prevent accidental triggering of the slave conversion. This prevents accidental triggering of the slave conversion. However, external triggering of both the master and slave ADCs must be activated at the same time.

There are 6 basic conversion modes:

- Simultaneous injection mode
- Simultaneous Rule Mode
- Fast crossover mode
- Slow crossover mode
- Alternate Trigger Mode
- Independent mode

It is also possible to use the above patterns in combination in the following ways:

- simultaneous injection mode + simultaneous rule mode
- Simultaneous Rule Mode + Alternate Trigger Mode
- simultaneous injection mode + crossover mode

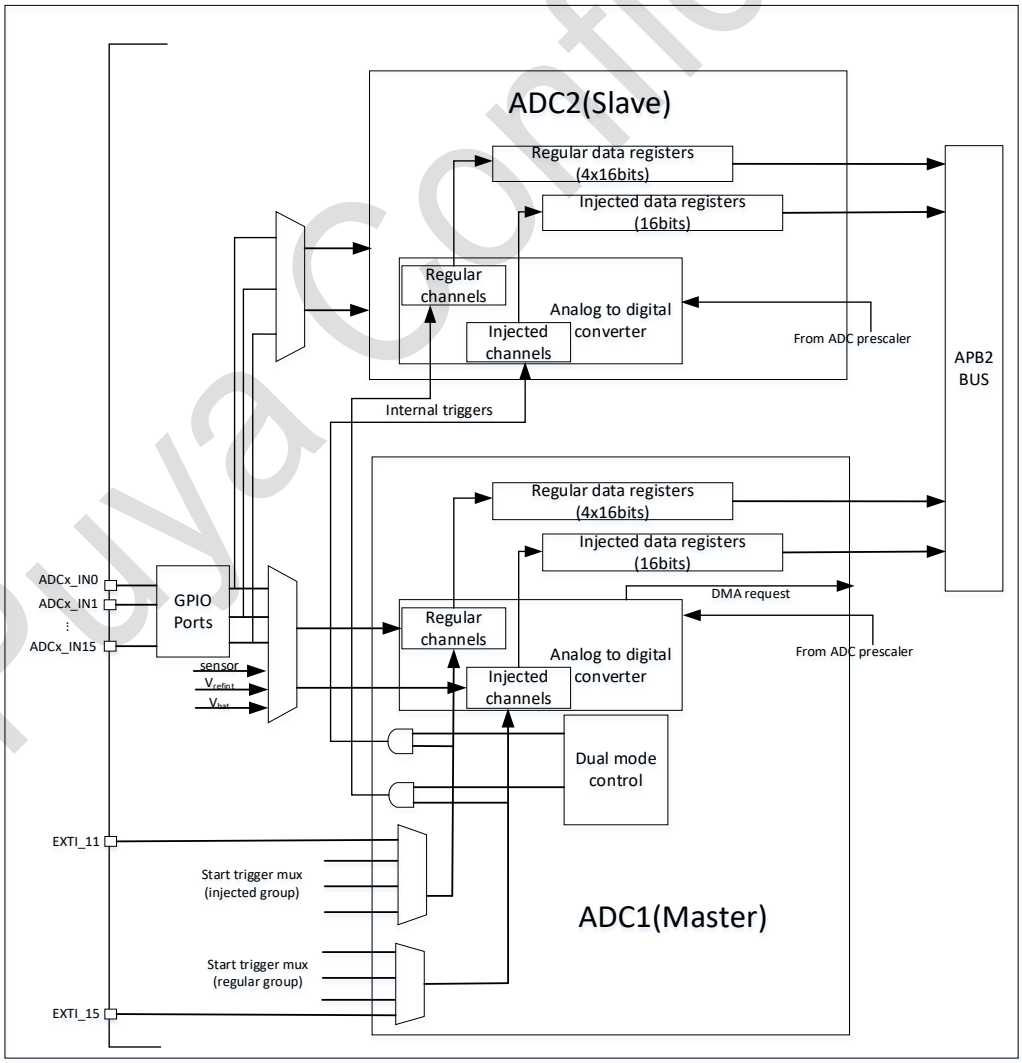


Figure 12-8 Dual ADC Module Diagram

1) An external trigger signal acts on ADC2 but is not shown in this diagram.
2) In some dual ADC modes, the rule conversion data for ADC1 and ADC2 is contained in the full ADC1 data register (ADC1_DR).

12.3.14.1. Synchronous Injection Mode

In synchronous injection mode, ADC1 and ADC2 synchronously convert groups of injection channels. the ADC_JSQR registers of ADC 1 and ADC 2 specify all the converted channels of an injection group sequence, where the JL[1:0] bits of the ADC_JSQR register specify the length of this sequence sequence

When the ADON bit is 1, ADC1 and ADC2 convert an injection sequence synchronously once the ADC1 external trigger (selected by JEXTSEL[2:0] of the ADC1_CR2 register) occurs.

Note:

1) Do not convert the same channel on two ADCs, i.e. the sampling time of two ADCs on the same channel should not overlap.

2) ADC1 and ADC2 synchronous conversion of the channel sampling time to maintain the same

3) At the end of the ADC1 or ADC2 conversion:

i. the converted data is stored in the ADC_JDRx register of each ADC.

ii. When both ADC1/ADC2 injection channels are converted, a JEOC interrupt is generated (if the interrupt is enabled by either ADC).

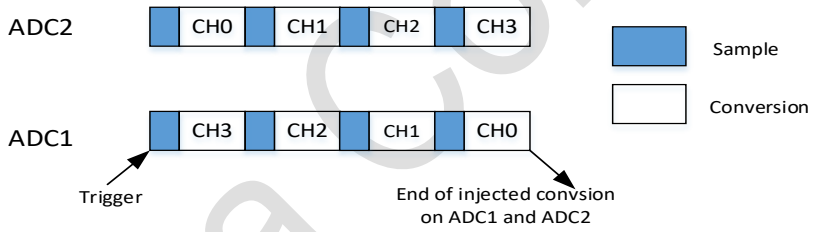


Figure 12-9 Synchronised injection pattern on 4 channels

12.3.14.2. Synchronisation rule model

In synchronous rule mode, ADC1 and ADC2 synchronously convert groups of rule channels. The ADC_SQRx registers of ADC 1 and ADC 2 specify all the converted channels of a rule sequence, where the L [1:0] bits of the ADC_SQR1 register specify the length of the sequence.

When the ADON bit is 1, ADC1 and ADC2 synchronise to convert a regular sequence as soon as the ADC1 external trigger (selected by EXTSEL[2:0] of the ADC1_CR2 register) occurs. (Note: users should not convert the same channel on 2 ADCs i.e. the sampling time of two ADCs on the same channel should not overlap).

At the end of ADC1 or ADC2 conversion:

- generates a 32-bit DMA transfer request (if the DMA bit is set) to the 32-bit ADC1_DR register in SRAM, the high half of the word contains the conversion data of ADC2 and the low half of the word contains the conversion data of ADC1.
- When both ADC1/ADC2 rule channels have been converted, an EOC interrupt is generated (if either ADC enables the interrupt).

Notes:

1) In synchronous rule mode, sequences of the same length must be converted or the trigger interval must be longer than the longer of the 2 sequences, otherwise the ADC conversion with the shorter sequence will be restarted while the conversion of the longer sequence has not completed. Otherwise, the ADC conversion with the shorter sequence will be restarted before the conversion of the longer sequence is completed.

2) ADC1 and ADC2 synchronised conversions should have the same channel sampling time.

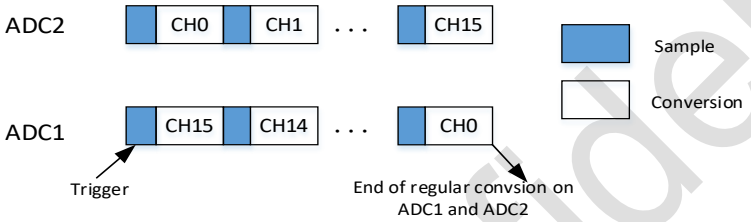


Figure 12-10 Synchronisation rule pattern on 16 channels

12.3.14.3. Rapid crossover mode

Fast crossover mode, where ADC1 and ADC2 convert one channel alternately. This mode is only applied to regular channel sets (usually one channel), and SQ1 of ADC_SQR3 of ADC 1 and ADC 2 defines the channel being converted.

When the ADON bit is 1, ADC1 and ADC2 will alternate converting a regular channel once the ADC1 external trigger (selected by EXTSEL [2:0] in the ADC1_CR2 register) occurs.

The external trigger source comes from ADC1's regular channel multiplexer. After the external trigger is generated:

- ADC2 starts immediately
- ADC1 starts after a delay of 7 ADC clock cycles

If the CONT bit of ADC1 and ADC2 are set at the same time, the two selected ADC rule channels will be converted consecutively. after ADC1 generates an EOC interrupt (enabled by EOCIE), a 32-bit DMA transfer request is generated (if the DMA bit is set), and 32-bit data from ADC1_DR register is transferred to SRAM. The 32-bit data in the ADC1_DR register is transferred to SRAM, and the high half of ADC1_DR contains the ADC2 conversion data and the low half contains the ADC1 conversion data.

Note: The maximum allowable sample time is <7 ADCCLK cycles to avoid the overlap of two sample cycles when ADC1 and ADC2 convert the same channel.

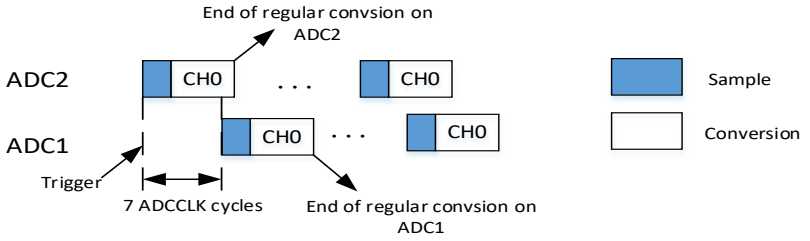


Figure 12-11 Fast crossover mode in continuous conversion mode on one channel

12.3.14.4. Slow crossover mode

This mode is only available for regular channel groups (one channel only). The external trigger source comes from the rule channel multiplexer of ADC1. After the external trigger is generated:

- ADC2 starts immediately
- ADC1 starts after a delay of 14 ADC clock cycles
- ADC2 starts again after a delay of the second 14 ADC cycles, and so on.

Note: The maximum allowable sample time is <14 ADCCLK cycles to avoid overlap with the next conversion.

After ADC1 generates an EOC interrupt (enabled by EOCIE), a 32-bit DMA transfer request is generated (if the DMA bit is set), and the 32-bit data in the ADC1_DR register is transferred to SRAM, with the high half of the ADC1_DR word containing ADC2's converted data, and the low half of the ADC1 word containing ADC1's converted data.

The CONT bit cannot be set in this mode because it will continuously convert the selected rule channel.

Note: The application must ensure that no external triggers are generated to inject channels when using crossover mode.

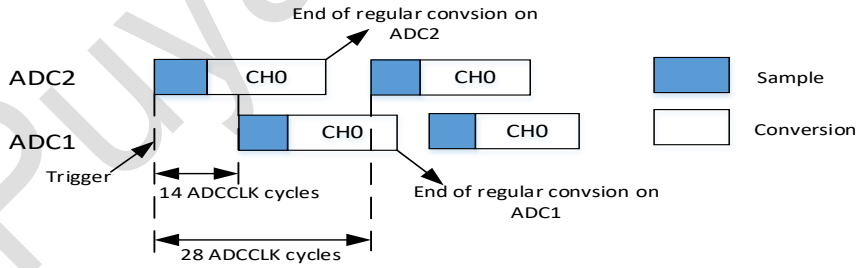


Figure 12-12 Slow crossover mode on one channel

12.3.14.5. Alternate trigger mode

This mode is only available for injection channel groups. The external trigger source comes from the injection channel multiplexer of ADC1 (SCAN mode).

- When the first trigger is generated, all injection group channels on ADC1 are converted.

- When the second trigger arrives, all injection group channels on ADC2 are converted.
- So looped .....

If JEOC interrupts are allowed, a JEOC interrupt is generated after all ADC1 injection group channels are converted.

If JEOC interrupts are allowed, a JEOC interrupt is generated after all ADC2 injection group channels are converted.

If another external trigger is generated after all injection group channels have been converted, alternate trigger processing restarts with the conversion of the ADC1 injection group channels.

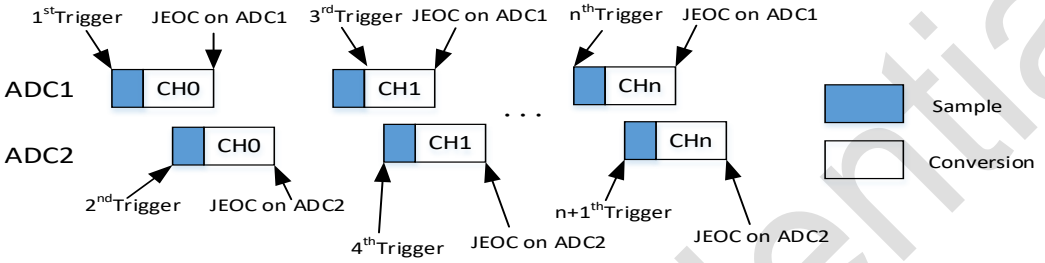


Figure 12-13Alternate Trigger - Group of Injection Channels per ADC

If injection interrupt mode is used on both ADC1 and ADC2:

- The first injection channel on ADC1 is converted when the first trigger is generated.
- When the second trigger arrives, the first injection channel on ADC2 is converted.
- so looped .....

If JEOC interrupts are allowed, a JEOC interrupt is generated after all ADC1 injection group channels are converted.

If JEOC interrupts are allowed, a JEOC interrupt is generated after all ADC2 injection group channels are converted.

When all injection group channels have converted, restart the alternate trigger process if another external trigger is generated.

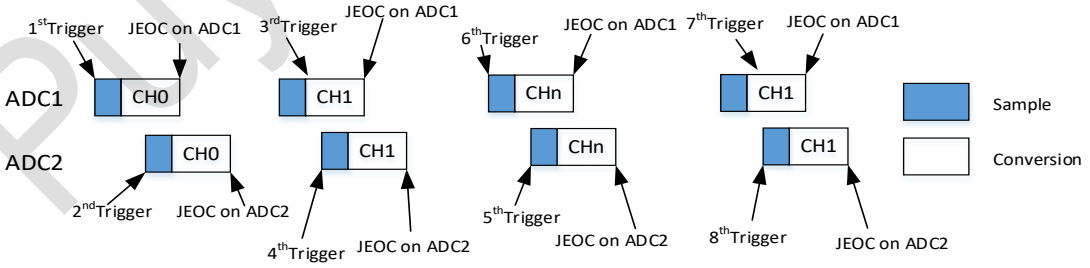


Figure 12-14Alternate triggering - 4 injection channels on each ADC in intermittent mode

12.3.14.6. Stand-alone model

In this mode, the dual ADCs operate asynchronously and each ADC operates independently.

12.3.14.7. Mixed rule/injection synchronisation patterns

Rule group synchronisation transitions can be interrupted to initiate synchronisation transitions for injected groups.

Notes:

1) In mixed rule/injection sync mode, sequences with the same time length must be converted or the trigger interval must be guaranteed to be longer than the longer of the 2 sequences, otherwise while the conversion of the longer sequence has not completed, the ADC conversion with the shorter sequence may be restarted when the conversion of the longer sequence has not been completed.
2) Synchronised ADC1 and ADC2 conversions with the same channel sampling time.

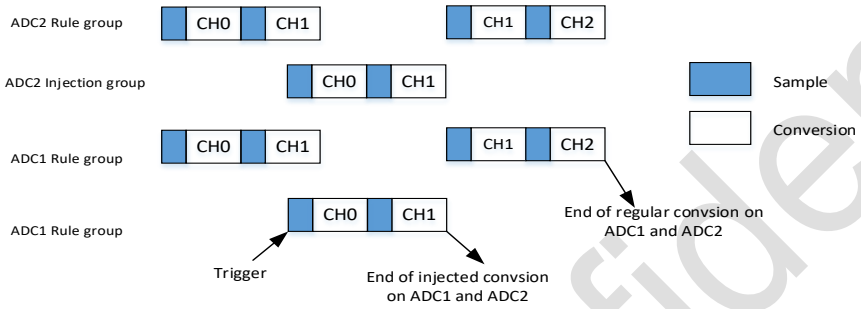


Figure 12-15Hybrid rule/injection synchronisation mode-scan mode

12.3.14.8. Hybrid synchronisation rules plus alternating trigger patterns

Rule group synchronisation transitions can be interrupted to initiate injection group alternate trigger transitions. Figure 4-17 shows a rule synchronisation transition interrupted by an alternation trigger.

An injection alternate transition is initiated immediately after the arrival of an injection event. If a rule conversion is already running, to ensure synchronisation after the injection conversion, rule conversions for all ADCs (master and slave) are stopped and synchronisation resumes at the end of the injection conversion.

Notes:

1) In mixed Sync Rule + Alternate Trigger mode, it is necessary to convert sequences with the same length or to ensure that the interval between triggers is longer than the longer of the 2 sequences, otherwise while the conversion of the longer sequence has not yet completed, the Otherwise the ADC conversion with the shorter sequence will be restarted when the conversion of the longer sequence has not been completed.

2) ADC1 and ADC2 synchronised conversions with the same sample time for the rule channel.

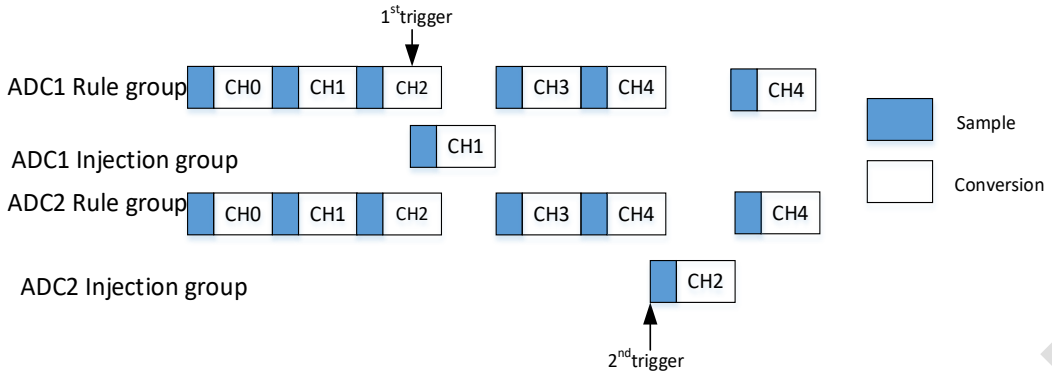


Figure 12-16Alternate plus rule synchronisation (with alternate trigger + jdiscen)

If the trigger event occurs during an injection transformation that interrupts a rule transformation, this trigger event is ignored. The following diagram shows the manipulation of this situation (the 2nd trigger is ignored)

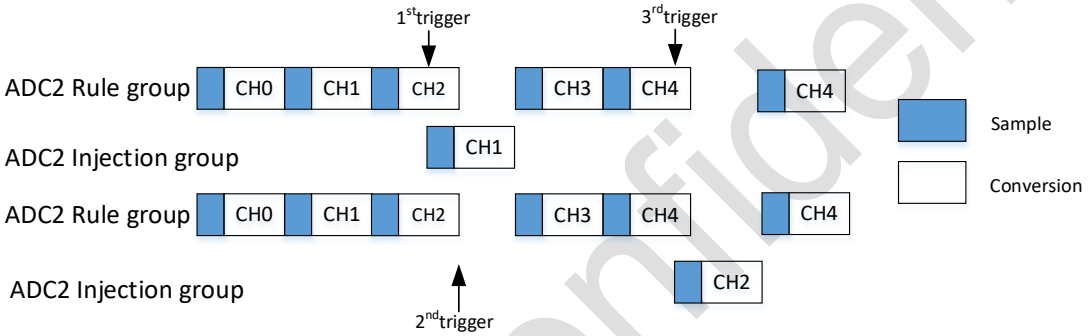


Figure 12-17 Trigger event occurs during injection transition

12.3.14.9. Mixed synchronous injection plus crossover mode

An injection event can interrupt a cross-conversion. In this case, the cross-conversion is interrupted, the injection conversion is initiated, and the cross-conversion is resumed at the end of the injection sequence conversion. The following figure shows an example of this situation.

Note: When the ADC clock prescaler factor is set to 4, the alternating mode does not evenly distribute the sample time, and the sample interval is 8 ADC clock cycles alternating with 6 ADC clock cycles instead of an even 7 ADC clock cycles.

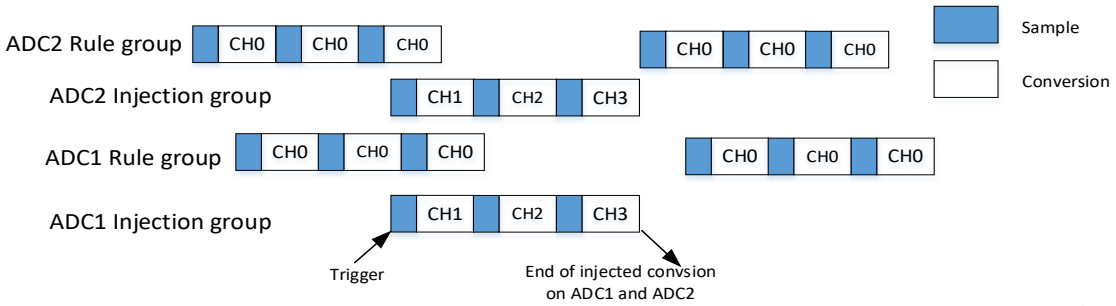


Figure 12-18 Single-channel conversion of injected sequence interrupt crossover (with simultaneous injection of interrupt cont + fast crossover)

12.3.15. Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the temperature (TA) around the device. The temperature sensor is internally connected to the ADC1_IN16 channel, which converts the sensor output voltage to a digital value. The recommended sampling time for the analogue input of the temperature sensor is 17.1 us. the sensor can be placed in power-down mode when not in use.

The output voltage of the temperature sensor varies linearly with temperature. Due to variations in the production process, the offset of the temperature variation curve may vary from chip to chip (up to 45°C difference). Internal temperature sensors are better suited for detecting changes in temperature rather than measuring absolute temperatures. If an accurate temperature measurement is required, an external temperature sensor should be used.

Note: The TSVREFE bit must be set to activate the conversion of the internal channels: ADC1_IN16 (temperature sensor) and ADC1_IN17 (VREFINT).

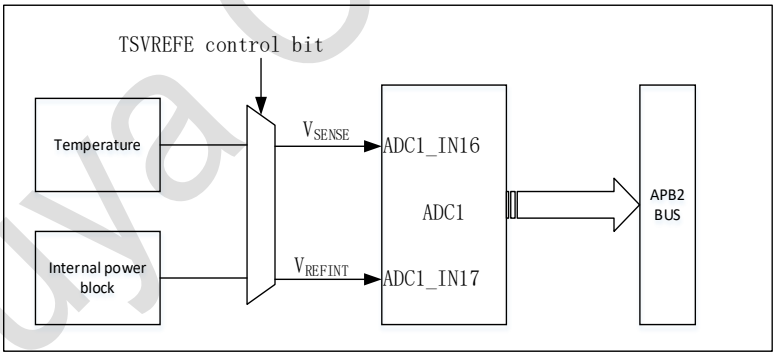


Figure 12-19 Temperature Sensor Channel Block Diagram

Key Features

Supported temperature range: -40 °C to 125 °C

Accuracy: ±1.5 °C

1) To read the temperature to use the sensor, do the following:

2) Select ADC1_IN16

3) Select a sample time that is greater than the minimum sample time specified in the datasheet.

4) Wake up the temperature sensor from power-down mode by placing TSVREFE position 1 in the ADC_CR2 register.

5) Start ADC conversion by placing ADON position 1 and enabling the external trigger

6) Read the VSENSE data generated in the ADC data registers

7) Calculate the temperature using the following formula:

Temperature (°C) = {(VSENSE - V25) / Avg_Slope} + 25

where:

V25 = VSENSE value at 25 °C

Avg_Slope = average slope of the temperature vs. VSENSE curve in mV/°C or μV/°C

(See the Electrical Characteristics section of the datasheet for information on actual V25 and Avg_Slope values.)

NOTE: Sensors require a setup time to wake up from power-down mode, after which the correct level of VSENSE is output. The ADC also requires a setup time after power-up, so to reduce the delay, both ADON and TSVREFE should be set to position 1 at the same time.

12.3.16. Battery monitoring

Since the VBAT voltage may be higher than the VDDA, the VBAT pin needs to be internally connected to the bridge distributor to ensure proper ADC operation.

The bridge is automatically enabled to connect VBAT/3 to the ADC1/2/3_IN18 input channel.

12.3.17. ADC interrupt

An interrupt can be generated by the occurrence of any of the following events: they all have independent interrupt enable bits.

End of rule group and injection group conversion;

- Analogue watchdog event;

There are 2 other flags present in the ADC_SR register, but they have no interrupt associated with them:

- JSTRT (start conversion of injector group channels)
- STRT (start converting channels of the rule group)

Note: ADC1 and ADC2 interrupts are mapped on the same interrupt vector, while ADC3 interrupts have their own interrupt vector.

Table 12-8 ADC Interrupts

Interrupt event	Event Flag	Enable Control Bit
End of rule group conversion	EOC	EOCIE
End of injection group transition	JEOC	JEOCIE
Analogue watchdog status bit set	AWD	AWDIE

12.4. ADC Register

ADC1 register base address: 0x4001_2400

ADC2 register base address: 0x4001_2800

ADC3 register base address: 0x4001_3C00

12.4.1. Status register (0x00: ADC_SR)

Address offset:0x00

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved											STRT	JSTRT	JEOC	EOC	AWD
RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RES.	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:5	Reserved	RES	27'h0	Reserved
4	STRT	RC_W0	0	Rule Channel Start Status Bit This bit is set to 1 by hardware at the start of the rule channel conversion and is set to 0 by a software write 0. 0: Rule channel conversion has not started 1: Rule channel conversion has started
3	JSTRT	RC_W0	0	Injection Channel Start Status Bit This bit is set 1 by hardware at the start of the injection channel group conversion, and is set 0 by software writing 0. 0: Injection channel conversion has not started 1: Injection channel conversion has started
2	JEOC	RC_W0	0	Injection Channel Conversion End Status Bit This bit is set by hardware to 1 at the end of conversion for all injected channel groups, and is set to 0 by software writing 0. 0: Conversion not completed 1: Conversion completed
1	EOC	RC_W0	0	End of Conversion Status Bit This bit is set 1 by hardware at the end of a (regular or injected) channel group conversion, set 0 by software writing 0 or set 0 when ADC_DR is read. 0: Conversion not completed 1: Conversion completed
0	AWD	RC_W0	0	Analogue Watchdog Flag Bit This bit is set 1 by hardware when the converted voltage value is outside the range defined by the ADC_LTR or below the ADC_HTR register, and is set 0 by software writing 0. 0: No analogue watchdog event occurred 1: Analogue watchdog event occurred

12.4.2. ADC Control Register 1 (0x04: ADC_CR1)

Address offset:0x04

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			Reserved	ADSTP	Reserved	RESSEL[1:0]		AWDEN	JAWDEN	Reserved		DUALMOD[3:0]			
RES.			RW	R_W1	RW	RW		RW	RW	RES.	RES.	RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
RW			RW	RW	RW	RW	RW	RW	RW	RW	RW				

Bit	Name	R/W	Reset Value	Function
31:29	Reserved	RES	4'h0	Reserved
28	Reserved	RES	1'h0	Reserved
27	ADSTP	R_W1	1'h0	ADC conversion stop enable. Software Write 1 set to 1. Hardware set to 0 when ADC stop signal is received. 1: Stop ADC conversion. 0: Do not stop ADC conversion.
26	Reserved	RES	1'h0	Reserved
25:24	RESSEL[1:0]	RW	2'b00	Resolution control bits. The resolution of the conversion can be selected by writing these bits in software. 00: 12 bits (16 ADCCLK cycles) 01: 10 bits (14 ADCCLK cycles) 10: 8 bits (12 ADCCLK cycles) 11: 6 bits (10 ADCCLK cycles)
23	AWDEN	RW	1'b0	Rule channel analogue watchdog enable. Enables the analogue watchdog on the rule channel. This bit is set by a software write 1 to 1 and a write 0 to 0. 0: Disable analogue watchdog on the rules channel 1: Use analogue watchdog on the rule channel
22	JAWDEN	RW	1'b0	Analogue watchdog enables on injection channel. Enables the analogue watchdog on the injection channel. This bit is set to 1 by a software write 1 and to 0 by a write 0. 0: Disable analogue watchdog on injection channel 1: Use analogue watchdog on injection channel
21:20	Reserved	RES	2'b0	Reserved
19:16	DUALMOD[3:0]	RW	4'h0	Dual mode selection control bits. These bits are reserved in ADC2 and ADC3; (In Dual ADC mode, changing the configuration of a channel produces a restart condition, which results in

Bit	Name	R/W	Reset Value	Function
				loss of synchronisation. It is recommended that dual mode be turned off before making any configuration changes.) 0000: Independent mode. 0001: Mixed sync rules + injected sync mode 0010: Mixed sync rules + alternate trigger mode 0011: Mixed sync injection + fast alternating mode 0100: Mixed sync injection + slow alternating mode 0101: Injection synchronisation mode 0110: Rule Synchronisation Mode 0111: Fast alternating mode 1000: Slow Alternating Mode 1001: Alternate Trigger Mode
15:13	DISCNUM[2:0]	RW	3'h0	Intermittent Mode Channel Count. The number of channels converted by the rule channel group after an external trigger is received in intermittent mode. A software write operation sets these bits. 000: 1 channel 001: 2 channels ..... 111: 8 channels
12	JDISCEN	RW	1'b0	Injection Channel Intermittent Mode Enable. This bit is set by a software write 1 to 1 and a write 0 to 0 to enable or disable intermittent mode on the injection channel group. 0: Intermittent mode is disabled on the injection channel group 1: Intermittent mode is used on the injection channel group
11	DISCEN	RW	1'b0	Intermittent Mode Enable for Rule Channel This bit is set by a software write 1 to 1 and a write 0 to 0 to enable or disable intermittent mode on the rule channel set 0: Intermittent mode is disabled on the rule channel group 1: interrupt mode is used on the rule channel group
10	JAUTO	RW	1'b0	Auto Inject Enable This bit is set by software write 1 to 1 and write 0 to 0 to enable or disable the automatic injection channel group conversion after the rule channel group conversion is finished 0: Disable automatic injection channel group conversion 1: Enable automatic injection channel group conversion
9	AWDSGL	RW	1'b0	Single Channel Watchdog Enable. This bit is set by a software write 1 to 1 and a write 0 to 0 to enable or disable the analogue watchdog on the channel defined by AWDCH[4:0]. 0: use analogue watchdog on all channels 1: Use analogue watchdog on a single channel
8	SCAN	RW	1'b0	Scan Mode Enable This bit is set by a software write 1 to 1 and a write 0 to 0 to enable or disable scan mode. In scan mode, the channel selected by the ADC_SQRx or ADC_JSQRx register is converted.

Bit	Name	R/W	Reset Value	Function
				0: Turn off scan mode 1: Use scan mode
7	JEOCIE	RW	1'b0	Inject Channel Conversion End Interrupt Enable This bit is set by software write 1 to set 1 and write 0 to set 0. When this bit is enabled, an interrupt request is generated while JEOC is active. 0: JEOC interrupt is disabled 1: JEOC interrupt is allowed.
6	AWDIE	RW	1'b0	Analogue Watchdog Interrupt Enable This bit is set by software write 1 to set 1 and write 0 to set 0. When this bit is enabled, an interrupt request is generated while AWD is active. 0: Analogue watchdog interrupt disabled 1: Analogue watchdog interrupt is allowed
5	EOCIE	RW	1'b0	Rule Channel Conversion End Interrupt Enable This bit is set by software write 1 to set 1 and write 0 to set 0. When this bit is enabled, an interrupt request is generated while EOC is active. 0: EOC interrupt is disabled 1: EOC interrupt is allowed.
4:0	AWDCH[4:0]	RW	5'h0	Analogue Watchdog Channel Selection Bit This bit is set by a software write and is used to select the input channel for the analogue watchdog. 00000: ADC analogue input channel 0 00001: ADC analogue input channel 1 ..... 01111: ADC analogue input channel 15 10000: ADC analogue input channel 16 10001: ADC analogue input channel 17 10010: ADC analogue input channel 18 10011: ADC analogue input channel 19 All other values are retained. Notes: - Analogue input channels 16 and 17 of ADC1 are connected to the temperature sensor and VREFINT respectively inside the chip, and ADC123 channel 18 is connected to Vbat. - Analogue input channels 16 and 17 of ADC23 are connected to VSS inside the chip.

12.4.3. ADC Control Register 2 (0x08: ADC_CR2)

Address offset:0x08

Reset value:0x0000_0000

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

Reserved							Reserved	TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2:0]			Reserved
RES.							RW	RW	R_W1	R_W1	RW	RW	RW	RW	RES.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
JEXTTRIG	JEXTSEL[2:0]			ALIGN	Reserved		DMA	Reserved				RSTCAL	CAL	CONT	ADON
RW	RW	RW	RW	RW	RES.		RW	RES.				R_W1	R_W1	RW	RW

Bit	Name	R/W	Reset Value	Function
31:25	Reserved	RES	7'h0	Reserved
24	Reserved	RES	1'b0	Reserved
23	TSVREFE	RW	1'b0	Temperature Sensor and VREFINT Enables Write 1 to set 1 and write 0 to set 0 via software to enable or disable the temperature sensor and VREFINT channels. Enableable in ADC1, reserved bits in ADC2 and ADC3. 0: Disable temperature sensor and VREFINT 1: Enable temperature sensor and VREFINT
22	SWSTART	R_W1	1'b0	Start conversion rule channel enable Conversion is initiated by a software write 1 set to 1, and is cleared by hardware to 0 immediately after the conversion is initiated. 0: Reset state 1: Start conversion rule channel
21	JSWSTART	RW	1'b0	Start conversion injection channel enable Conversion is initiated by software writing 1 to set 1, and is cleared by hardware to set 0 immediately after the conversion is initiated. 0: Reset state 1: Start conversion of injection channel
20	EXTTRIG	RW	1'b0	Rule Channel External Trigger Enable This bit is cleared by the software Write 1 Set 1 and Write 0 Set 0, and is used to enable or disable external trigger signals that can initiate rule channel group conversions. 0: Disable rule channel external trigger 1: Enable rule channel external trigger
19:17	EXTSEL[2:0]	RW	3'b000	Rule Channel External Trigger Event Selection Bit. Selects the external event that initiates the rule channel group conversion ADC1, ADC2 external trigger events are as follows: 000: CH1 event for timer 1 001: CH2 event of Timer1 010: CH3 event of timer 1 011: CH2 event of timer 2 100: TRGO event of timer 3

Bit	Name	R/W	Reset Value	Function
				101: CH4 event of timer 4 110: TRGO event for timer 8 or EXTI11 (selected via ADC1_ETRGREG_REMAP, ADC2_ETRGREG_REMAP of SYSCFG module) 111: SWSTART The ADC3 external trigger is configured as follows: 000: CH1 event of timer 3 001: CH3 event for timer 2 010: CH3 event of timer 1 011: CH1 event of timer 8 100: TRGO event of timer 8 101: CH1 event of timer 5 110: CH3 event of timer 5 111: SWSTART
16	Reserved	RES	1'b0	Reserved
15	JEXTTRIG	RW	1'b0	Inject channel external trigger enable 0: Disable injection channel external trigger 1: Enable injection channel external trigger
14:12	JEXTSEL[2:0]	RW	3'h0	Injecting the Channel Group External Trigger Event Selection Bit The external trigger events for ADC1 and ADC2 are as follows: 000: TRGO event for timer 1 001: CH4 event for timer 1 010: TRGO event of timer 2 011: CH1 event of timer 2 100: CH4 event of timer 3 101: TRGO event for timer 4 110: CH4 event for Timer 8 or EXTI15 (selected via ADC1_ETRGINJ_REMAP, ADC2_ETRGINJ_REMAP of SYSCFG module) 111: JSWSTART The trigger settings for ADC3 are as follows: 000: TRGO event of timer 1 001: CH4 event of timer 1 010: CH3 event of timer 4 011: CH2 event of timer 8 100: CH4 event of timer 8 101: TRGO event of timer 5 110: CH4 event of timer 5 111: JSWSTART
11	ALIGN	RW	1'b0	Data Alignment Control Bit This bit is cleared by software write 1 set 1 and write 0 set 0. 0: Right alignment 1: Left Alignment
10:9	Reserved	RES	2'h0	Reserved
8	DMA	RW	1'b0	DMA Enable Bit

Bit	Name	R/W	Reset Value	Function
				This bit is cleared by software write 1 set 1 and write 0 set 0. 0: Disable DMA mode 1: Enable DMA mode Note: In dual adc mode, only ADC1 can generate DMA requests.
7:4	Reserved	RES	4'h0	Reserved
3	RSTCAL	R_W1	1'b0	Calibration Reset Enable Bit This bit is set by a software write 1 to 1 and cleared by hardware to 0. This bit is cleared after the calibration register has been initialised (i.e. after RSTCAL is set to 1). 0: Calibration register is initialised 1: The calibration register is initialised Note: When a conversion is in progress, clearing the calibration register requires an additional cycle if RSTCAL is set.
2	CAL	R_W1	1'b0	Calibration Enable This bit is set by software write 1 to start calibration and cleared by hardware on calibration failure or calibration success. When ADON is invalid and SWSTART, JSWSTART is invalid; software starts calibration. 0: Calibration complete 1: enable calibration
1	CONT	RW	1'b0	Continuous Conversion Enable This bit is cleared by software write 1 set 1 and write 0 set 0. If this bit is set, conversions are performed continuously until the bit is cleared. 0: Single conversion mode 1: Continuous conversion mode
0	ADON	RW	1'b0	On/Off A/D Converter ADC converter work enable, when set to 1 after the ADC converter wake up, in the converter power-up to start conversion there is a delay tSTAB. when set to 0 after the ADC converter is in power-down state. 0: Disable ADC conversion, ADC converter enters power-down mode. 1: Enable ADC converter. Note: If SWSTART,JSWSTART is changed in this register along with ADON, the conversion is not triggered. This is to prevent triggering a wrong conversion.

12.4.4. ADC Sampling Time Register 1 (0x0C: ADC_SMPR1)

Address offset:0x0C

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	

RES.								RW		RW		RW			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
RW	RW			RW			RW			RW			RW		

Bit	Name	R/W	Reset Value	Function
31:24	Reserved	RES.	8'h0-	Reserved
23:0	SMPx[2:0]	RW	24'h0	Selecting the Sample Time for Channel x Setting these bits via software is used to independently select the sample time for each channel. The channel selection bits must remain unchanged during the sampling period. 000: 3.5 cycles 100: 28.5 cycles 001: 5.5 cycles 101: 41.5 cycles 010: 7.5 cycles 110: 134.5 cycles 011: 13.5 cycles 111: 239.5 cycles

12.4.5. ADC Sampling Time Register 2 (0x10: ADC_SMPR2)

Address offset:0x10

Reset value:0x0000_0000

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
RES.		RW			RW			RW			RW			RW	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
RW	RW			RW			RW			RW			RW		

Bit	Name	R/W	Reset Value	Function
31:30	Reserved	RES	2'h0-	Reserved
29:0	SMPx[2:0]	RW	30'h0	Selecting the Sample Time for Channel x Setting these bits via software is used to independently select the sample time for each channel. The channel selection bits must remain unchanged during the sampling period. 000: 3.5 cycles 100: 28.5 cycles 001: 5.5 cycles 101: 41.5 cycles 010: 7.5 cycles 110: 134.5 cycles 011: 13.5 cycles 111: 239.5 cycles

12.4.6. ADC Injection Channel Data Offset Register x (0x14-0x20: ADC_JOFRx)

x=1~4

Address offset:0x14~0x20

Reset value:0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	20'h0	Reserved
11:0	JOFFSETx[11:0]	RW	12'h0	Inject channel xth conversion data offset Software configures the value of these bits. When the conversion is injected into the channel, these bits define the value used to subtract from the original conversion data. The final conversion result can be read out in the ADC_JDRx register.

12.4.7. ADC Watchdog High Threshold Register (0x24: ADC_HTR)

Address offset:0x24

Reset value:0x0000_0FFF

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	20'h0	Reserved
11:0	HT[11:0]	RW	12'hFFF	Analogue Watchdog High Threshold Software configures the value of these bits. These bits define the analogue watchdog threshold high limit.

12.4.8. ADC Watchdog Low Threshold Register (0x28: ADC_LTR)

Address offset:0x28

Reset value:0x0000_0000

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

Reserved	LT[11:0]
RES.	RW

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	20'h0-	Reserved
11:0	LT[11:0]	RW	0x000	Analogue Watchdog Low Threshold Software configures the value of these bits. These bits define the analogue watchdog threshold low limit.

12.4.9. ADC Rule Sequence Register 1 (0x2C: ADC_SQR1)

Address offset:0x2C

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
RES.								RW				RW			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SQ16[0]	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
RW	RW					RW					RW				

Bit	Name	R/W	Reset Value	Function
31:24	Reserved	RES	8'h0-	Reserved
23:20	L[3:0]	RW	4'b0000	Rule Channel Sequence Length The software configures the value of these bits. These bits define the number of channels in the rule channel conversion sequence. 0000: 1 conversion 0001: 2 conversions ..... 1111: 16 conversions
19:15	SQ16[4:0]	RW	5'b00000	The software configures the value of these bits. 16th Conversion in Rule Sequence These bits define the number of the 16th conversion channel (0 to 19) in the conversion sequence.
14:10	SQ15[4:0]	RW	5'b00000	Software configures the value of these bits. The 15th conversion in the rule sequence, these bits define the number of the 15th conversion channel in the conversion sequence (0~19).
9:5	SQ14[4:0]	RW	5'b00000	Software configures the value of these bits. The 14th conversion in the rule sequence, these bits define the number of the 14th conversion channel in the conversion sequence (0~19).
4:0	SQ13[4:0]	RW	5'b00000	Software configures the value of these bits. The 13th conversion in the rule sequence, these bits define the number of the 13th conversion channel in the conversion sequence (0~19).

12.4.10. ADC Rule Sequence Register 2 (0x30: ADC_SQR2)

Address offset:0x030

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
RES.		RW					RW					RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10[0]	SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
RW	RW					RW					RW				

Bit	Name	R/W	Reset Value	Function
31:30	Reserved	RES	2'h0	Reserved
29:25	SQ12[4:0]	RW	5'b00000	The software configures the value of these bits. 12th Conversion in Rule Sequence These bits define the number of the 12th conversion channel (0 to 19) in the conversion sequence.
24:20	SQ11[4:0]	RW	5'b00000	Software configures the value of these bits. The 11th conversion in the rule sequence, these bits define the number (0~19) of the 11th conversion channel in the conversion sequence.
19:15	SQ10[4:0]	RW	5'b00000	Software configures the value of these bits. The 10th conversion in the rule sequence, these bits define the number of the 10th conversion channel in the conversion sequence (0~19).
14:10	SQ9[4:0]	RW	5'b00000	Software configures the value of these bits. 9th conversion in the rule sequence, these bits define the number (0~19) of the 9th conversion channel in the conversion sequence.
9:5	SQ8[4:0]	RW	5'b00000	Software configures the value of these bits. The 8th conversion in the rule sequence, these bits define the number of the 8th conversion channel in the conversion sequence (0~19).
4:0	SQ7[4:0]	RW	5'b00000	Software configures the value of these bits. The 7th conversion in the rule sequence, these bits define the number of the 7th conversion channel in the conversion sequence (0~19).

12.4.11. ADC Rule Sequence Register 3 (0x34: ADC_SQR3)

Address offset:0x34

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
RES.		RW					RW					RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[0]	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
RW	RW					RW					RW				

Bit	Name	R/W	Reset Value	Function
31:30	Reserved	RES	2'h0	Reserved
29:25	SQ6[4:0]	RW	5'b00000	The software configures the value of these bits. 6th Conversion in Rule Sequence These bits define the number of the 6th conversion channel (0 to 19) in the conversion sequence.
24:20	SQ5[4:0]	RW	5'b00000	Software configures the value of these bits. The 5th conversion in the rule sequence, these bits define the number (0~19) of the 5th conversion channel in the conversion sequence.
19:15	SQ4[4:0]	RW	5'b00000	Software configures the value of these bits. 4th conversion in the rule sequence, these bits define the number of the 4th conversion channel in the conversion sequence (0~19).
14:10	SQ3[4:0]	RW	5'b00000	Software configures the value of these bits. The 3rd conversion in the rule sequence, these bits define the number (0~19) of the 3rd conversion channel in the conversion sequence.
9:5	SQ2[4:0]	RW	5'b00000	Software configures the value of these bits. The 2nd conversion in the rule sequence, these bits define the number of the 2nd conversion channel in the conversion sequence (0~19).
4:0	SQ1[4:0]	RW	5'b00000	Software configures the value of these bits. The 1st conversion in the rule sequence, these bits define the number of the 1st conversion channel in the conversion sequence (0~19).

12.4.12. ADC Injection Sequence Register (0x38: ADC_JSQR)

Address offset:0x38

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										JL[1:0]		JSQ4[4:1]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES.										RW		RW			
JSQ4[0]	JSQ3[4:0]					JSQ2[4:0]				JSQ1[4:0]					
RW	RW					RW				RW					

Bit	Name	R/W	Reset Value	Function
31:22	Reserved	RES	10'h0	Reserved
21:20	JL[1:0]	RW	2'b00	Injecting Channel Sequence Length The software configures the value of these bits. These bits define the number of transitions in the injection channel transition sequence. 00: 1 conversion 01: 2 conversions 10: 3 conversions 11: 4 conversions
19:15	JSQ4[4:0]	RW	5'b00000	Inject the 4th transition in the sequence The software configures the value of these bits. These bits define the number (0 to 19) of the 4th conversion channel in the conversion sequence.

Bit	Name	R/W	Reset Value	Function
				Note: Unlike regular conversion sequences, if the length of JL[1:0] is less than 4, the sequence order of conversions begins with (4-JL). For example, ADC_JSQR[21:0] = 10 00011 00011 00111 00010 means that the scan conversions will be performed in the following channel Sequential conversion: 7, 3, 3 instead of 2, 7, 3
14:10	JSQ3[4:0]	RW	5'b00000	Software configures the value of these bits. Inject the 3rd transition in the sequence
9:5	JSQ2[4:0]	RW	5'b00000	Software configures the value of these bits. Inject the 2nd conversion in the sequence
4:0	JSQ1[4:0]	RW	5'b00000	Software configures the value of these bits. Inject the 1st conversion in the sequence

12.4.13. ADC Injection Data Register x (0x3C-0x48: ADC_JDRx) x=1~4

Address offset:0x3C~0x48

Reset value:0x0000_0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	RES	16'h0	Reserved
15:0	JDATA[15:0]	R	16'h0	Inject channel conversion results The value of these bits is readable by the software. These bits are read-only and contain the conversion results for the injection channel. The data is left- or right-aligned

12.4.14. ADC Rules Data Register (0x4C: ADC_DR)

Address offset:0x4C

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2DATA[15:0]															
R															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
R															

Bit	Name	R/W	Reset Value	Function
31:16	ADC2DATA[15:0]	R	16'h0	Data converted by ADC2

Bit	Name	R/W	Reset Value	Function
				Software can read the value of these bits. - In ADC1: Dual mode, these bits contain the rule channel data converted by ADC2. - In ADC2: Reserved bits.
15:0	DATA[15:0]	R	16'h0	Rule channel conversion results The software can read the value of these bits. These bits are read-only and contain the result of the rule channel conversion. Data is left or right aligned

12.4.15. ADC Calibration Configuration and Status Registers (0x50: ADC_CCSR)

Address offset:0x50

Reset value:0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALON	CAPSUC	OFFSUC	Reserved												
R	RC_W1	RES.													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	Reserved	CALSMP		CALSEL	Reserved										
RW	RW	RW	RW	RW	RES.										

Bit	Name	R/W	Reset Value	Function
31	CALON	R	1'b0	Calibration status bits. 1: ADC calibration is in progress; 0: ADC calibration is finished or not started.
30	CAPSUC	RC_W1	1'b0	Capacitor Calibration Status Bit. Indicates whether ADC capacitance calibration is successful. Hardware set to 1; software write 1 set to 0; CALON=0, CALSEL=0, CALSUC=1: Invalid state CALON=0, CALSEL=0, CALSUC=0: CAPs calibration not performed CALON=0, CALSEL=1, CALSUC =1: ADC CAPs calibration successful CALON=0, CALSEL=1, CALSUC =0: ADC CAPs calibration failed
29	OFFSUC	RC_W1	1'b0	Offset calibration status bit. Indicates whether ADC offset calibration is successful. Hardware set to 1; software writes 1 to 0; CALON=0, CALSEL=0,OFFSUC=0: ADC OFFSET calibration failed. CALON=0, CALSEL=0, OFFSUC=1: ADC OFFSET calibration successful CALON=0, CALSEL=1,OFFSUC=1: ADC OFFSET calibration successful CALON=0, CALSEL=1, OFFSUC=0: ADC OFFSET calibration failed

Bit	Name	R/W	Reset Value	Function
28:16	Reserved	RES.	13'h0	
15	Reserved	RES	1'h0	Reserved
14	Reserved	RES	1'h0	Reserved
13:12	CALSMP	RW	2'h0	Calibration Sample Time Selection. When CAL is 0, the software sets the selection. Configure the number of clock cycles for the calibration sampling phase based on the following information: 00: 1 ADC clock cycle 01: 2 ADC clock cycles 10: 4 ADC clock cycles 11: 8 ADC clock cycles The longer the period of the SMP is configured during calibration, the more accurate the calibration results will be, but this configuration will result in a longer calibration period
11	CALSEL	RW	1'h0	Calibration content selection. When CAL is 0, the software setup selects the content to be calibrated. 1: Calibrate OFFSET and capacitor. 0: Calibrate OFFSET only
10:0	Reserved	RES.	11'h0	

12.4.16. ADC Register Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT	JSTRT	JEOC	EOC	AWD
	Reset value																												0	0	0	0	0
0x04	ADC_CR1	Res.	Res.	Res.	Res.	ADSTP	Res.	RESSEL[1:0]		AWDEN	JAWDEN	Res.	Res.	DUALMODE[3:0]				DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
	Reset value							0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	ADC_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2:0]			Res.	JEXTTRIG	JEXTSEL[2:0]			ALIGN	Res.	Res.	DMA	Res.	Res.	Res.	Res.	RSTCAL	CAL	CONT	ADON
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0			0					0	0	0	0
0x0C	ADC_SMPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMP17[2:0]			SMP16[2:0]			SMP15[2:0]			SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	ADC_SMPR2	Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]			SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	ADC_JOFR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JOFFSET1[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	ADC_JOFR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JOFFSET2[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x1C	ADC_JOFR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JOFFSET3[11:0]											

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x20	ADC_JOFR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JOFFSET4[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x24	ADC_HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x28	ADC_LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x2C	ADC_SQR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L[3:0]				SQ16[4:0]					SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	ADC_SQR2	Res.	Res.	SQ12[4:0]					SQ11[4:0]					SQ10[4:0]					SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	ADC_SQR3	Res.	Res.	SQ6[4:0]					SQ5[4:0]					SQ4[4:0]					SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	ADC_JSQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JL[1:0]		JSQ4[4:0]					JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]				
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	ADC_JDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	ADC_JDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	ADC_JDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	ADC_JDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA[15:0]																

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	ADC_DR	ADC2DATA[15:0]																DATA[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	ADC_CCSR	CALON	CAPSUC	OFFSUC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALSMP		CALSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0																0	0	0											

13. Advanced timers (TIM1 and TIM8)

13.1. Introduction

The Advanced Control Timer (TIM1 and TIM8) consists of a 16-bit auto-load counter driven by a programmable prescaler.

It is suitable for a variety of purposes, including measuring the pulse width of an input signal (input capture), or generating an output waveform (output compare, PWM, complementary PWM with embedded dead time).

Adjustment of pulse width and waveform period from a few microseconds to a few milliseconds is possible using the timer prescaler and the RCC clock control prescaler.

The advanced control timers (TIM1 and TIM8) and the general-purpose timer (Timex) are completely independent; they do not share any resources.

13.1.1. TIM1 and TIM8 main features

TIM1 and TIM8 timer functions include:

- 16-bit up, down, and up/down auto-load counters
- 16-bit programmable (can be modified in real time) prescaler, with a dividing factor of any value between 1 and 65536 for the counter clock frequency
- Up to 4 independent channels:
  - Input capture
  - Output comparison
  - PWM generation (edge or centre aligned mode)
  - Single pulse mode output
- Complementary outputs with programmable dead time
- Synchronisation circuitry to control timer-to-timer interconnections via external signals
- Repeat counter that allows the timer register to be updated after a specified number of counter cycles
- Brake input signal that places the timer output signal in a reset state or a known state
- Interrupt/DMA is generated when the following events occur:

  Update: counter overflow up/down, counter initialisation (via software or internal/external trigger)
  - Trigger event (counter start, stop, initialisation or count triggered internally/externally)
  - Input capture
  - Output comparison
  - Brake signal input
- Supports incremental (quadrature) encoder and Hall sensor circuits for positioning.
- Trigger input as external clock or per-cycle current management

13.1.2. Module Block Diagram

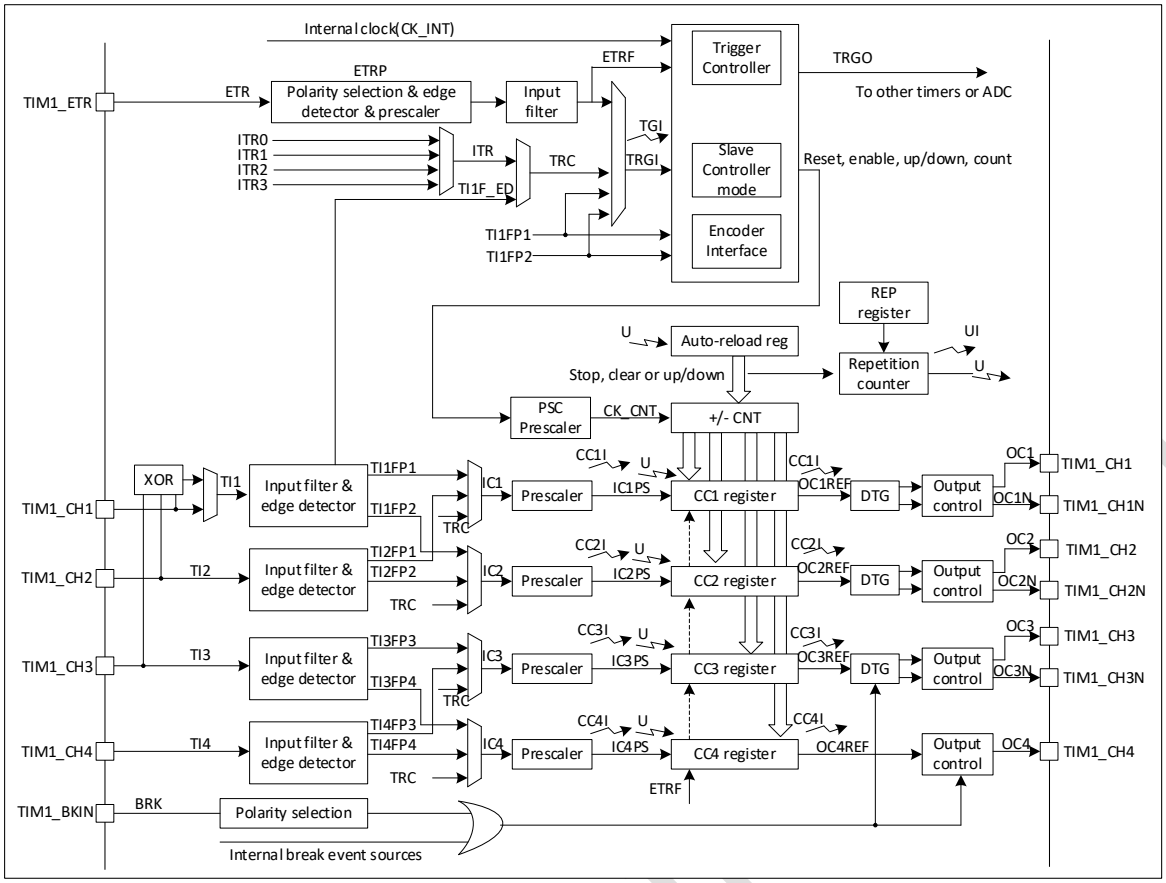


Figure 13-1 TIM1 and TIM8 modules

13.2. TIM1 and TIM8 Functional Descriptions

13.2.1. Time base unit

The main part of the programmable advanced control timer is a 16-bit counter and its associated auto-load register. This counter can count up, count down, or count up and down in both directions. The counter clock can be divided by a prescaler.

The counter, the autoload register and the prescaler register can be read and written by software, and the reads and writes remain valid even when the counter is running.

The time base unit contains:

- Counter register (TIMx_CNT)
- prescaler register (TIMx_PSC)
- Autoload Register (TIMx_ARR)
- Repeat Count Register (TIMx_RCR)

The autoload register is preloaded, and writing or reading the auto-reload register will access its preloaded register. Depending on the setting of the Auto-Load Pre-Load Enable bit (ARPE) in the TIMx_CR1 register, the contents of the Pre-Load Register are transferred to the Shadow Register either immediately or at each Update Event (UEV). An update event is generated when the counter reaches an overflow condition (overflow or underflow) and when the UDIS bit in the TIMx_CR1

register is equal to zero. Update events can also be generated by software and other conditions. The generation of update events for each configuration is described in detail later.

The counter is driven by the prescaler-divided clock output CK_CNT, which is valid for the counter only when the counter enable bit (CEN) in the TIMx_CR1 register is set. (See the description of the Slave Mode Controller for more details on enabling the counter).

Note that the counter starts counting one clock cycle after the CEN bit of the TIMx_CR register is set.

13.2.1.1. Prescaler description

The prescaler divides the counter clock frequency by any value between 1 and 65536. It is based on a 16-bit counter controlled by a 16-bit register (in the TIMx_PSC register). Because this control register has a buffer, it can be changed at runtime. The new prescaler parameter will be used when the next update event arrives.

Figures 14-2 and 14-3 give examples of changing counter parameters while the prescaler is running.

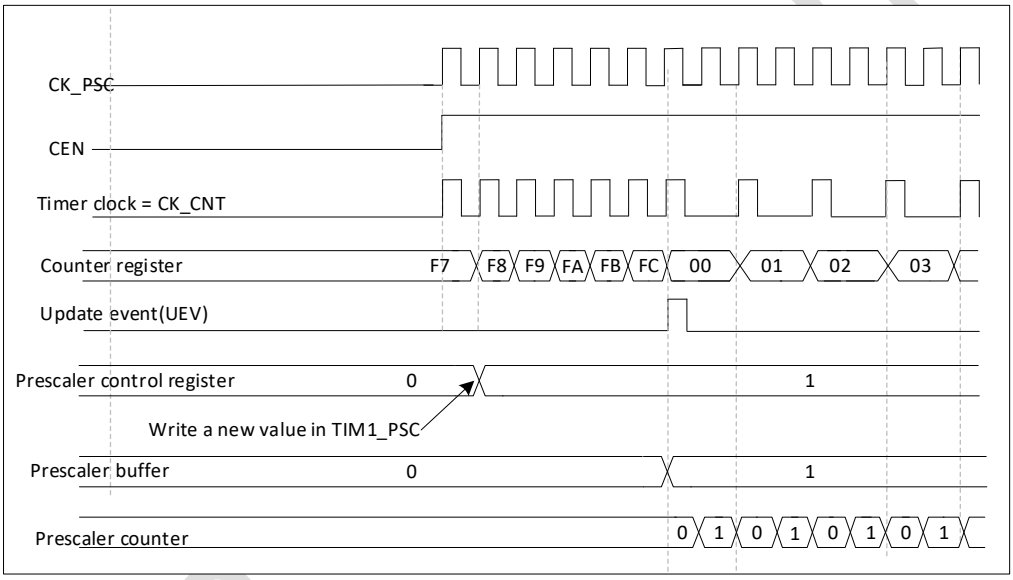


Figure 13-2 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 2

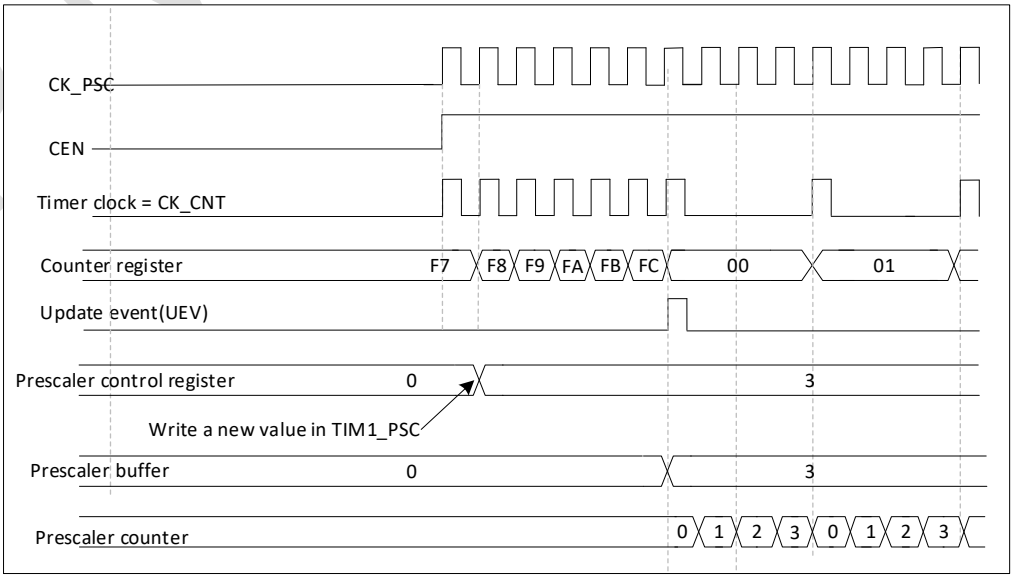


Figure 13-3 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 4

13.2.2. Counter mode

13.2.2.1. Upward count mode

In up-count mode, the counter counts from 0 to the auto-load value (the contents of TIMx_ARR), then starts counting from 0 again and generates a count overflow event.

If a repeat counter is used, then the update event (UEV) needs to be generated when the number of overflows reaches the value of the configured repeat counter register plus one (i.e., TIMx_RCR+1); if no repeat counter is used (i.e., TIMx_RCR=0), then the update event is generated every time the count overflows.

And setting the UG bit in the TIMx_EGR register (either by software or by using a slave mode controller) can similarly generate an update event.

The update event can be disabled by setting the UDIS bit in the TIMx_CR1 register; this prevents the shadow register from being updated when a new value is written to the preload register. An update event will not be generated until the UDIS bit is cleared '0'. Even then, the counter will still be cleared '0' when an update event should be generated, and the counter inside the prescaler will also be cleared '0' (but the prescaler value will remain unchanged).

In addition, if the URS bit (Select Update Request Source) in the TIMx_CR1 register is set, an update event UEV can be generated by setting the UG bit, but the UIF flag bit will not be set (i.e. no interrupt or DMA request will be generated). This is to avoid clearing the counter on a capture event and generating both update and capture interrupts.

When an update event occurs, all of the following registers are updated and the hardware simultaneously (based on the URS bit) sets the update flag bit (the UIF bit in the TIMx_SR register):

- The repeat counter is reloaded with the contents of the TIMx_RCR register.
- The autoload shadow register is reloaded with the value of the preload register (TIMx_ARR).
- The prescaler buffer is set to the value of the preload register (the contents of the TIMx_PSC register).

The following figure gives some examples of the action of the counter at different clock frequencies when TIMx_ARR = 0x36.

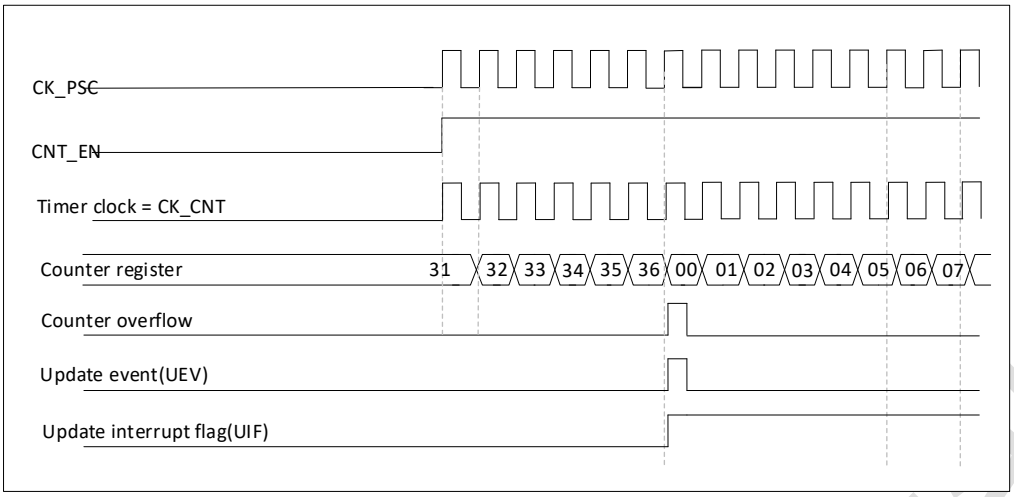


Figure 13-4 Counter Timing Diagram with Internal Clock Division Factor of 1

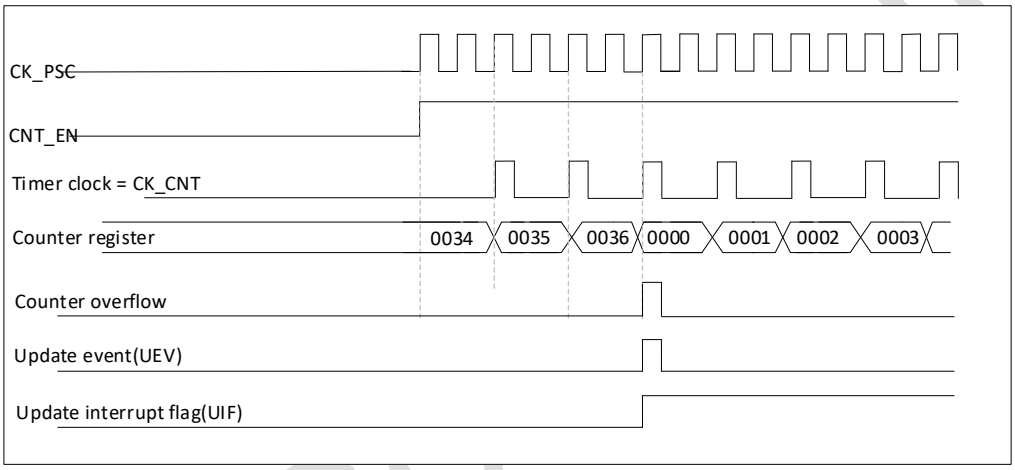


Figure 13-5 Counter Timing Diagram with Internal Clock Division Factor of 2

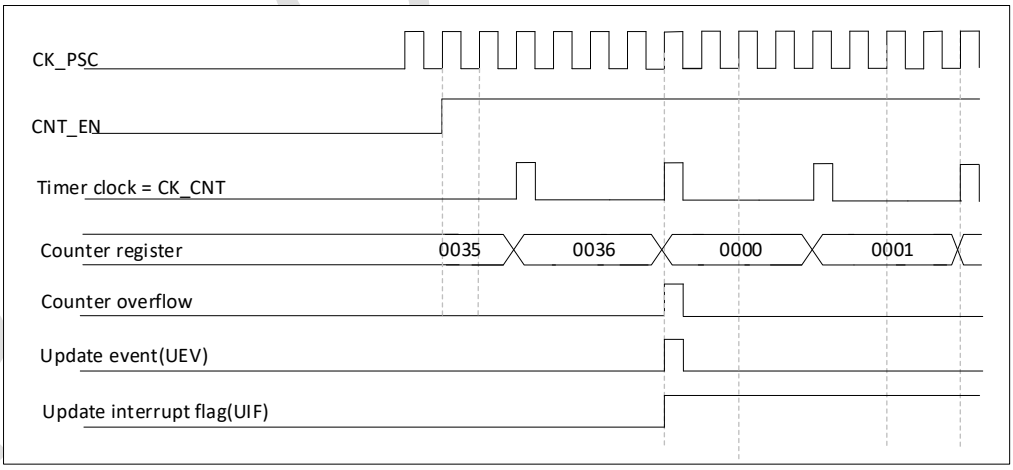


Figure 13-6 Counter Timing Diagram with Internal Clock Division Factor of 4

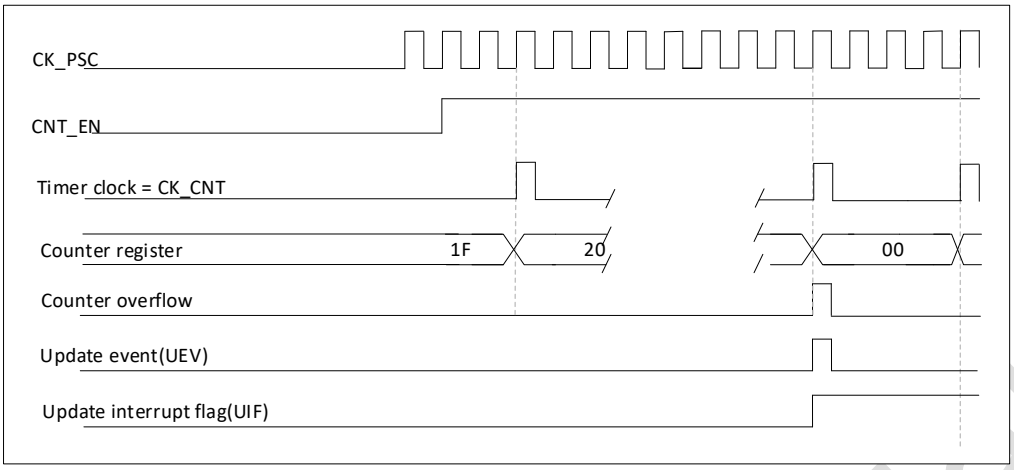


Figure 13-7 Timing diagram of the counter with internal clock division factor N

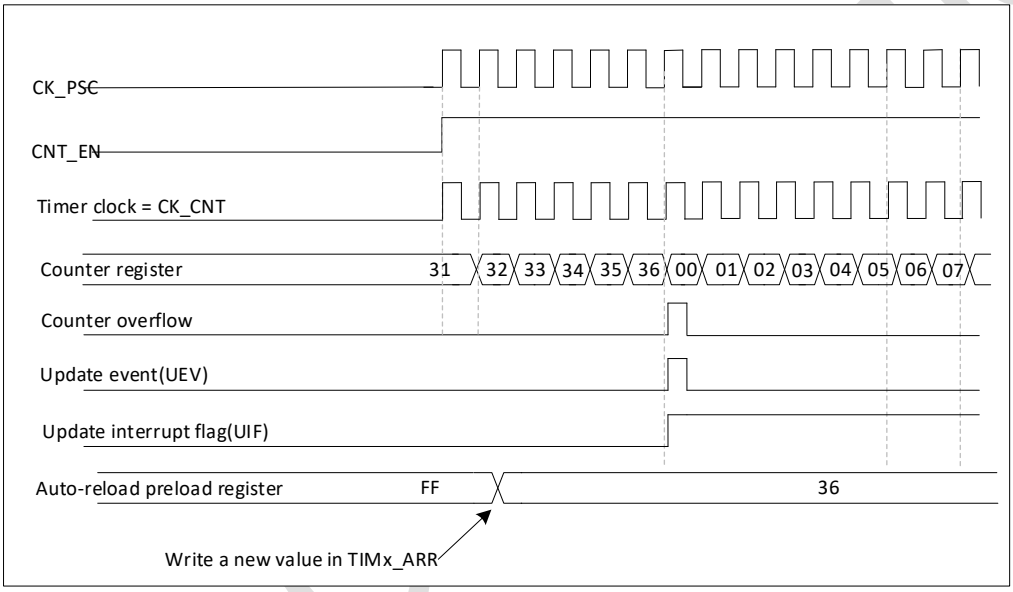


Figure 13-8 Counter timing diagram, update event when ARPE = 0 (no TIMx_ARR preloaded)

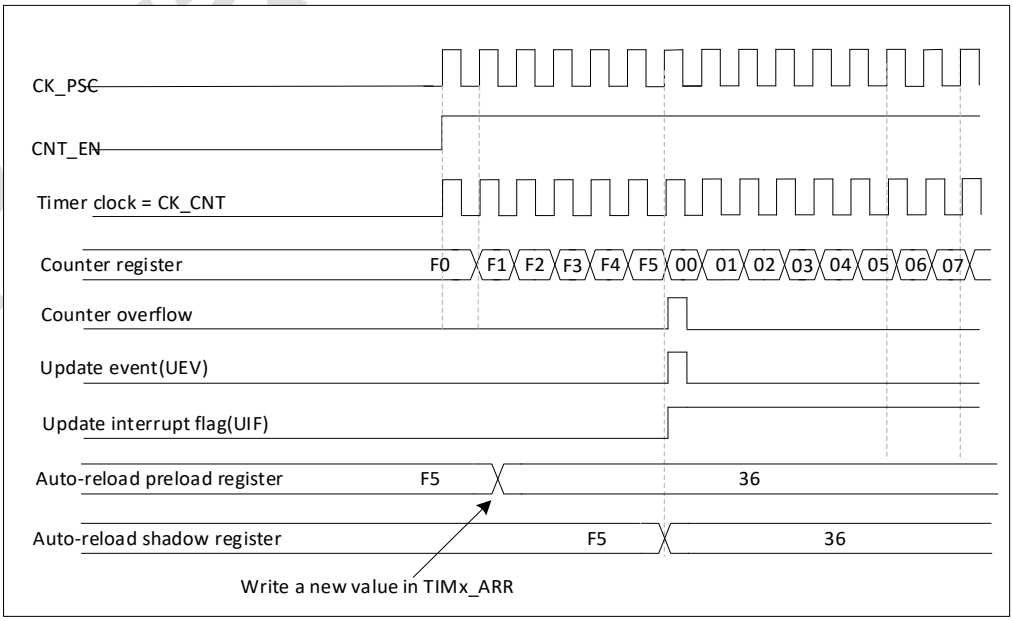


Figure 13-9 Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)

13.2.2.2. Downward count mode

In down count mode, the counter counts down from the auto-loaded value (the contents of TIMx_ARR) to 0, then restarts from the auto-loaded value and generates a count underflow event.

If a repeat counter is used, then the update event (UEV) is generated only when the number of underflows reaches the value of the configured repeat counter register plus one (i.e., TIMx_RCR+1); if no repeat counter is used (i.e., TIMx_RCR=0), then an update event is generated every time the count overflows.

And setting the UG bit in the TIMx_EGR register (either by software or by using a slave mode controller) can similarly generate an update event.

The update event can be disabled by setting the UDIS bit in the TIMx_CR1 register; this prevents the shadow register from being updated when a new value is written to the preload register. An update event is not generated until the UDIS bit is cleared '0'. Even then, when an update event should be generated, the counter will still restart counting from the current autoload value while the counter inside the prescaler is cleared '0' (but the prescaler coefficient remains unchanged).

In addition, if the URS bit (Select Update Request Source) in the TIMx_CR1 register is set, an update event UEV can be generated by setting the UG bit without setting up the UIF flag bit (i.e., no interrupt or DMA request will be generated), this is to avoid clearing the counters when a capture event occurs, and generating both update and capture interrupts.

When an update event occurs, all following registers are updated and the hardware simultaneously (based on the URS bit) sets the update flag bit (UIF bit in the TIMx_SR register):

- The repeat counter is reloaded with the contents of the TIMx_RCR register.
- The prescaler buffer is set to the value of the preload register (contents of the TIMx_PSC register).
- The current autoload register is updated with the value of the preload register (contents of the TIMx_ARR register).

Note: The autoload value is updated before the counter is reloaded, so the next cycle will be the expected value.

Here are some examples of counter operation at different clock frequencies when TIMx_ARR=0x36.

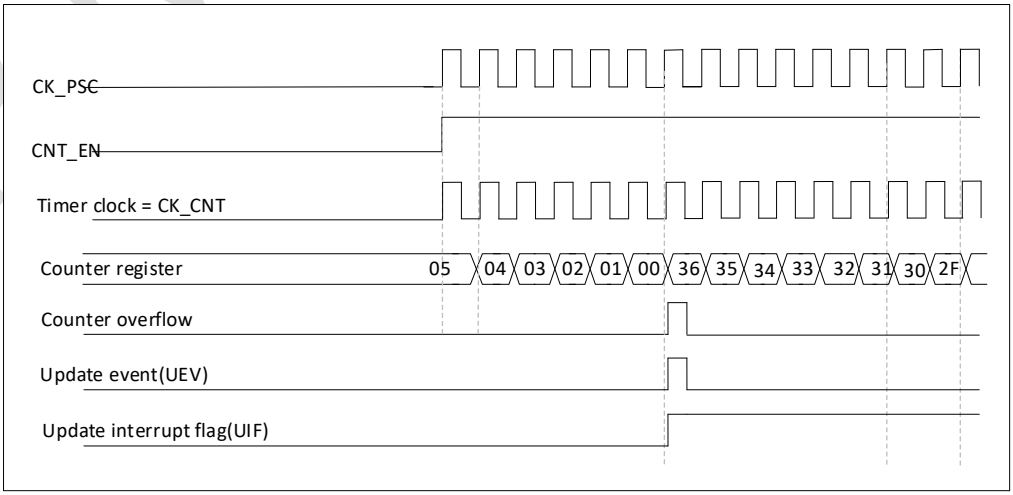


Figure 13-10 Counter Timing Diagram with Internal Clock Division Factor of 1

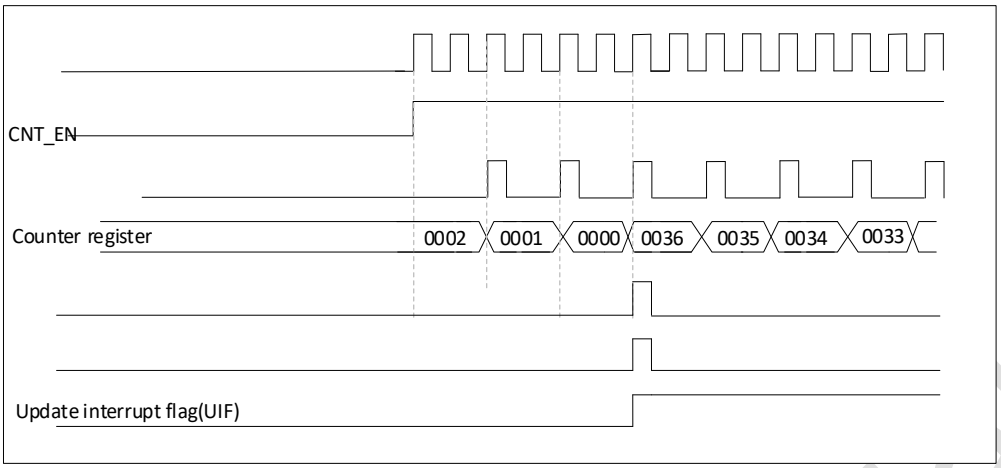


Figure 13-11 Counter Timing Diagram with Internal Clock Division Factor of 2

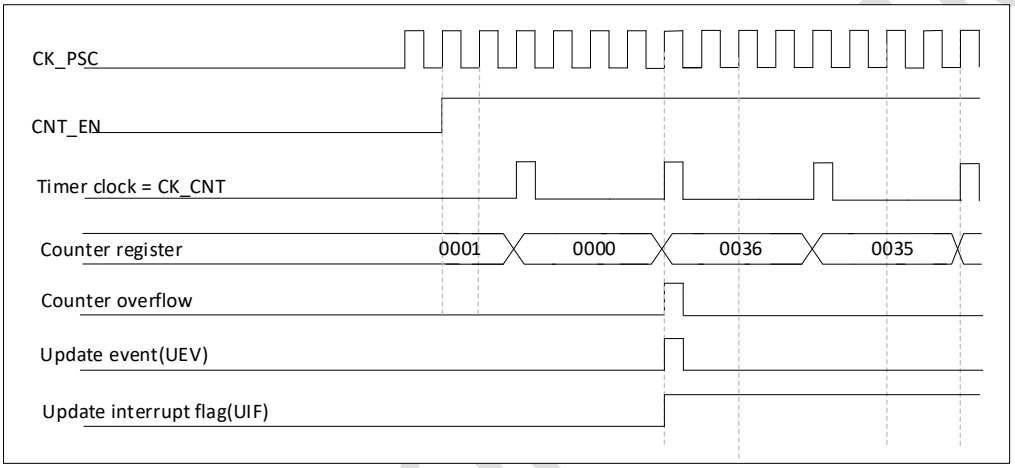


Figure 13-12 Counter Timing Diagram with Internal Clock Division Factor of 4

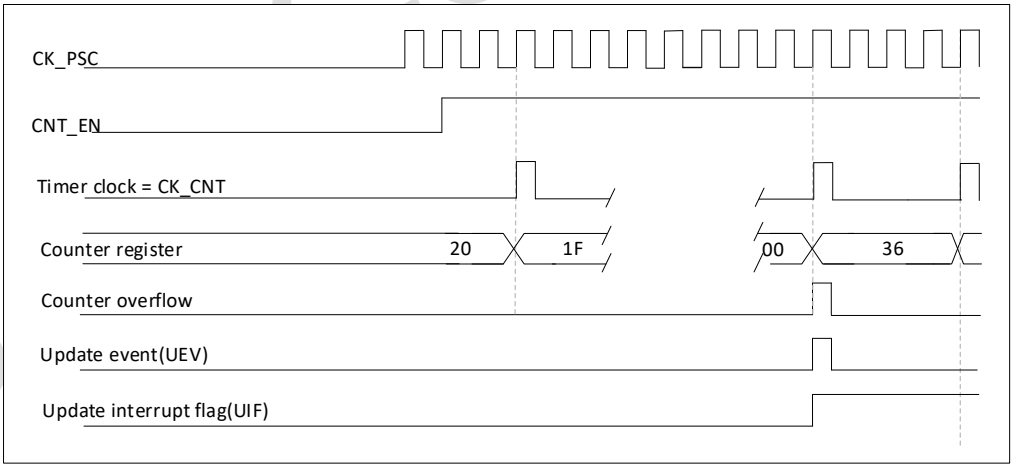


Figure 13-13 Timing diagram of the counter with internal clock division factor N

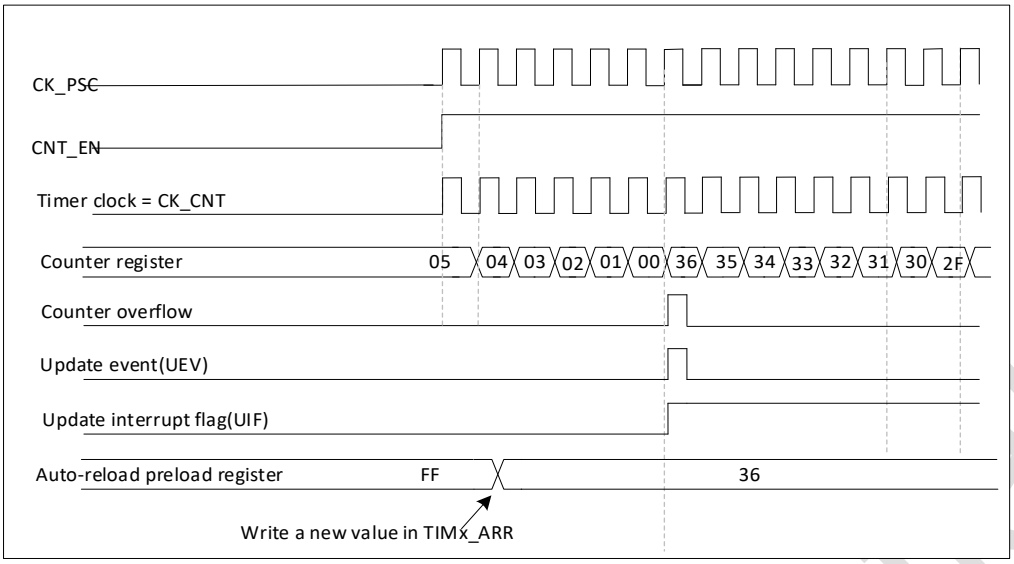


Figure 13-14 Counter Timing Diagram, Update Event When Repeat Counter Not Used

13.2.2.3. Central alignment mode (counting up/down)

In central alignment mode, the counter counts from 0 to the auto-loaded value (TIMx_ARR register) - 1, generates a counter overflow event, then counts down to 1 and generates a counter underflow event; and then counts from 0 again.

Central alignment mode can be obtained by configuring the CMS bit in the TIMx_CR1 register not to be '00'. The Output Compare Flag bit with the channel configured for Output Mode is set during the following types of counting: down counting (Central Alignment Mode 1, CMS='01'), up counting (Central Alignment Mode 2, CMS='10'), when counting up and down (Central Alignment Mode 3, CMS='11').

In this mode, the DIR direction bit in TIMx_CR1 cannot be written. It is updated by hardware and indicates the current count direction.

An update event can be generated on every count overflow and every count underflow; it can also be generated by setting (in software or using a slave mode controller) the UG bit in the TIMx_EGR register. The counter then resumes counting from 0, and the presale's internal counter resumes counting from 0 as well.

Setting the UDIS bit in the TIMx_CR1 register disables the update event. This prevents the shadow register from being updated when a new value is written to the preload register. Although the update event is not generated until the UDIS bit is cleared to 0. However, the counter will still continue to count up or down depending on the current auto-reload value.

In addition, if the URS bit (Select Update Request Source) in the TIMx_CR1 register is set, an update event UEV will be generated by setting the UG bit but not setting the UIF flag (and therefore not generating an interrupt and DMA request), this is to avoid clearing the counter when a capture event occurs and generating both an update and capture interrupt.

When an update event occurs, all registers are updated and (depending on the setting of the URS bit) the update flag bit (the UIF bit in the TIMx_SR register) is also set:

- Repeat counter is reset to the contents of the TIMx_RCR registers
- The prescaler buffer is loaded to the preloaded (TIMx_PSC register) value.

- The current autoload register is updated to the preloaded value (contents of the TIMx_ARR register).

Note: If an update occurs because of a counter overflow, the auto-reload will be updated before the counter is reloaded, so the next cycle will be the expected value (the counter is loaded with the new value).

Here are some examples of counter operation at different clock frequencies:

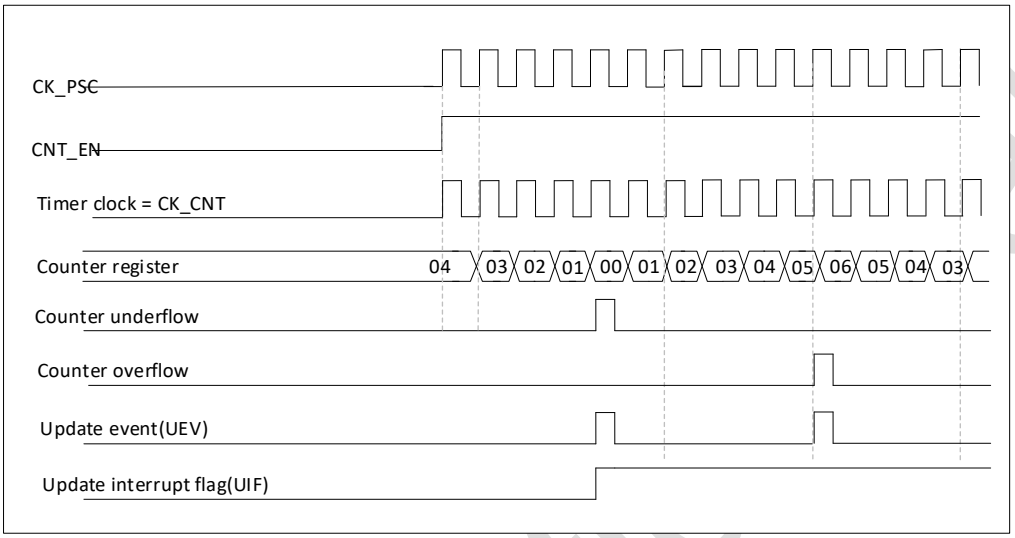


Figure 13-15 Counter Timing Diagram with Internal Clock Division Factor of 1, TIMx_ARR=0x6

Centre alignment mode 1 is used here.

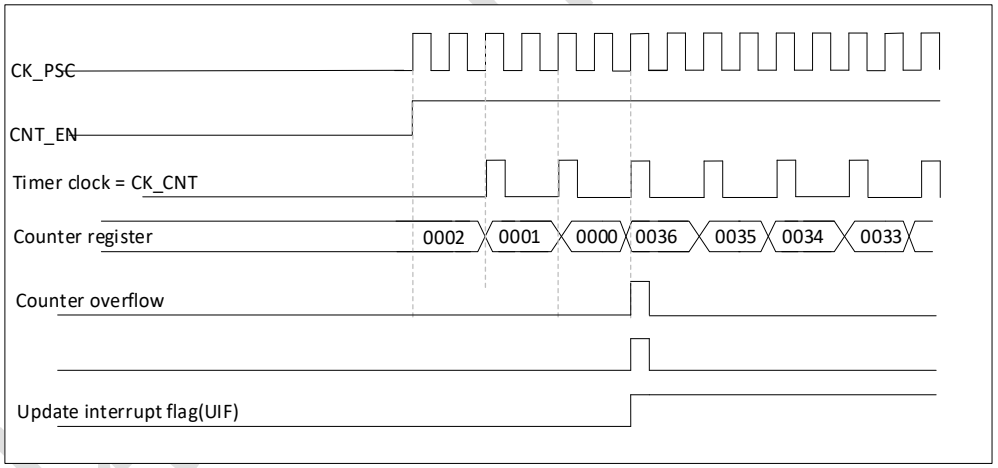


Figure 13-16 Counter Timing Diagram with Internal Clock Division Factor of 2

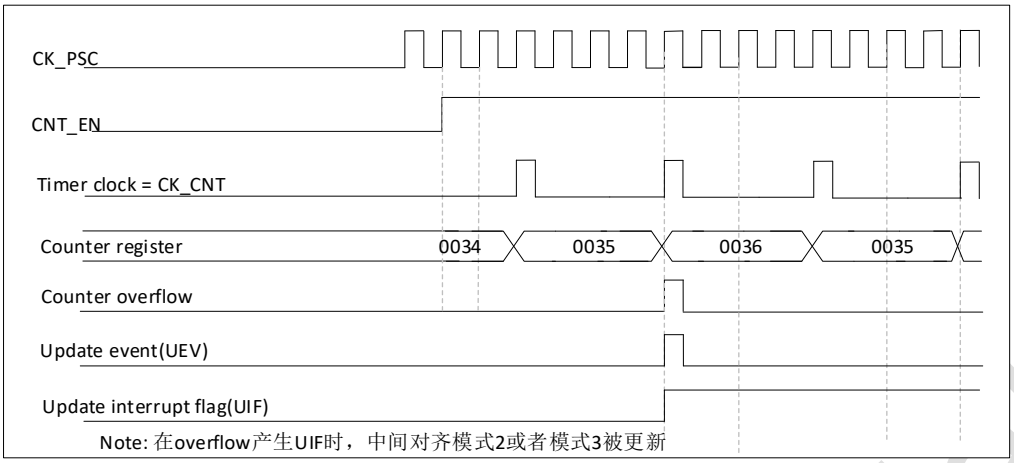


Figure 13-17 Counter Timing Diagram with Internal Clock Division Factor of 4, TIMx_ARR=0x36

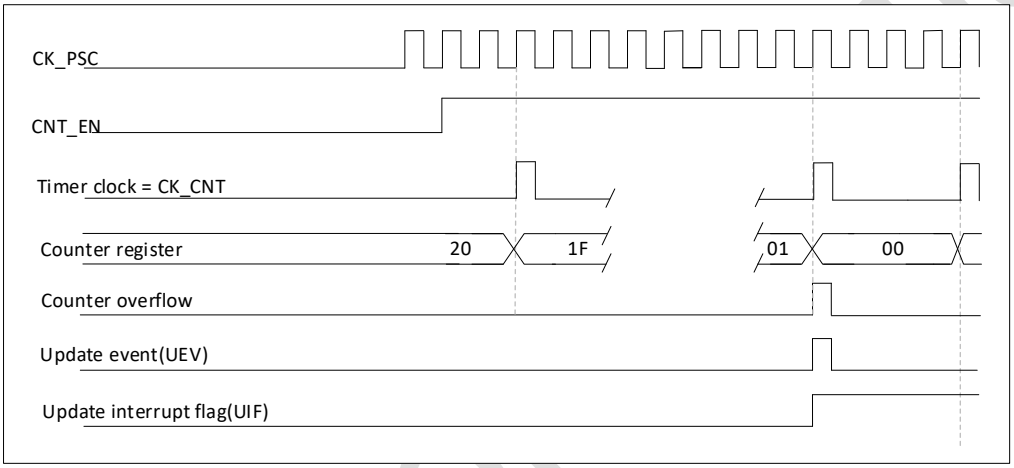


Figure 13-18 Timing diagram of the counter with internal clock division factor N

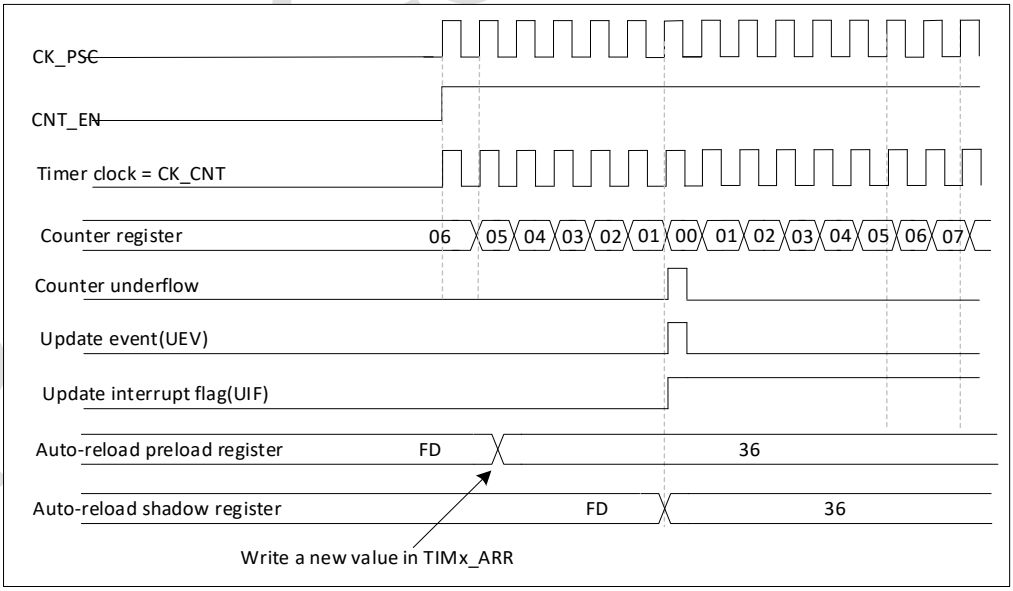


Figure 13-19 Counter Timing Diagram, Update Event at ARPE=1 (Counter Underflow)

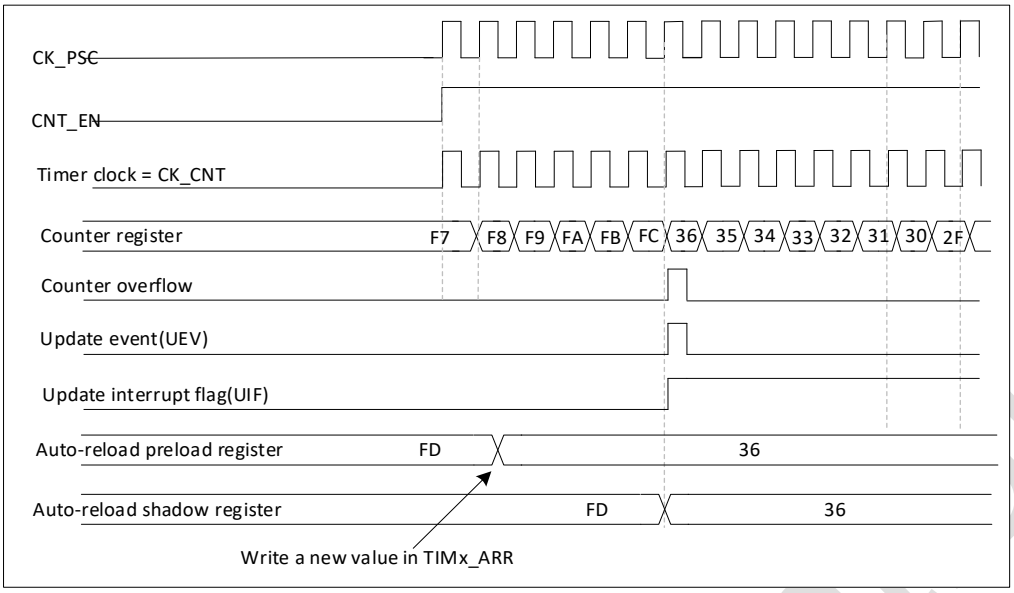


Figure 13-20 Counter Timing Diagram, Update Event at ARPE=1 (Counter Overflow)

13.2.3. Repetition counter

The "Time Base Unit" explains how the Update Event on Counter Overflow/Underflow (UEV) is generated, and in fact it can only be generated when the Repeat Counter reaches zero. This feature is very useful for generating PWM signals.

This means that data is only transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC preload register, and also the capture/compare register TIMx_CCRx in compare mode) every N+1 counts of overflows or underflows, and N is the value in the TIMx_RCR repeat counter register.

The repeat counter is decremented when either of the following conditions hold:

- Each time the counter overflows in upward counting mode
- Each time the counter underflows in downward counting mode.
- for every upflow and every downflow in central alignment mode.

Although this limits the maximum cycle period of the PWM to 128, it is able to update the duty cycle 2 times per PWM cycle. In centre-aligned mode, because the waveform is symmetrical, the maximum resolution is 2xTck if the compare register is only refreshed once in each PWM cycle.

The repeat counter is automatically reloaded and the repeat rate is defined by the value of the TIMx_RCR register. When an update event is generated by software (by setting the UG bit in TIMx_EGR) or by the hardware slave mode controller, the update event occurs immediately and the contents of the TIMx_RCR register are reloaded into the repeat counter, regardless of the value of the repeat counter.

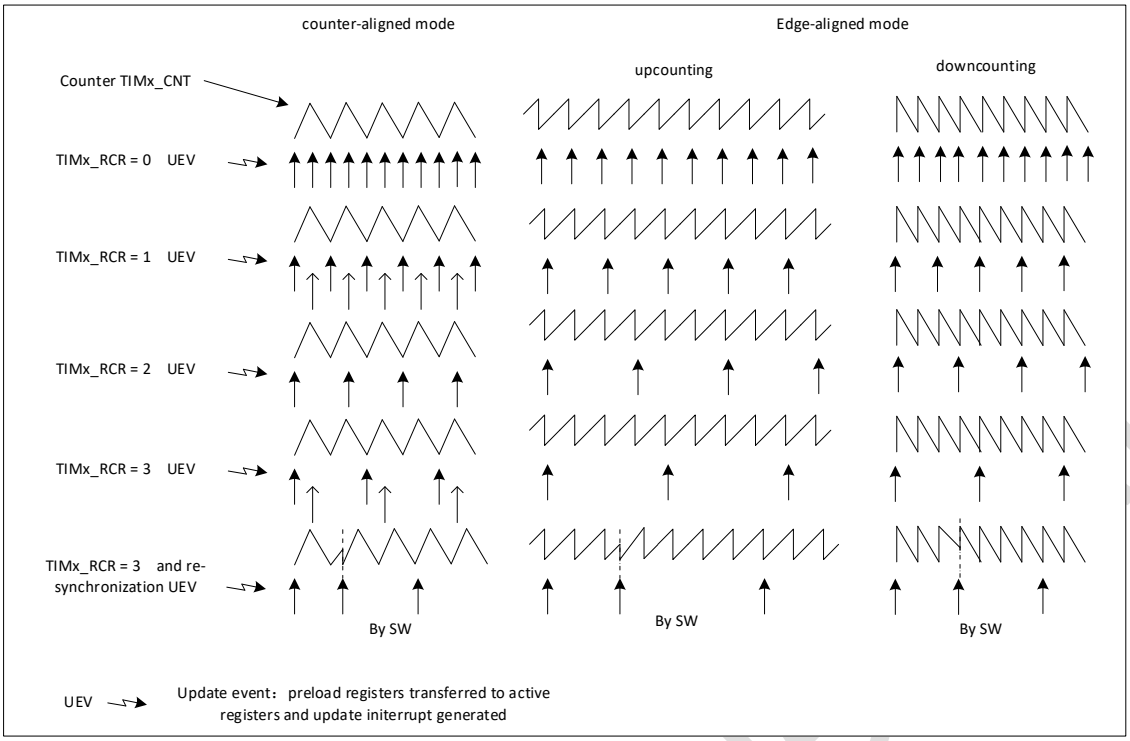


Figure 13-21 Examples of update rates in different modes, and register settings for TIMx_RCR

13.2.4. Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode 1: external input pin
- External clock mode 2: external trigger input ETR
- Internal Trigger Input (ITRx): uses one timer as a prescaler for another timer. For example, you can configure one timer Timer1 as a prescaler for another timer Timer2.

13.2.4.1. Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), the CEN, DIR (TIMx_CR1 register) and UG bits (TIMx_EGR register) are de facto control bits and can only be modified by software (the UG bit is still cleared automatically). As long as the CEN bit is written to '1', the prescaler clock is provided by the internal clock CK_INT.

The following diagram shows the operation of the control circuit and up counter in general mode without prescaler.

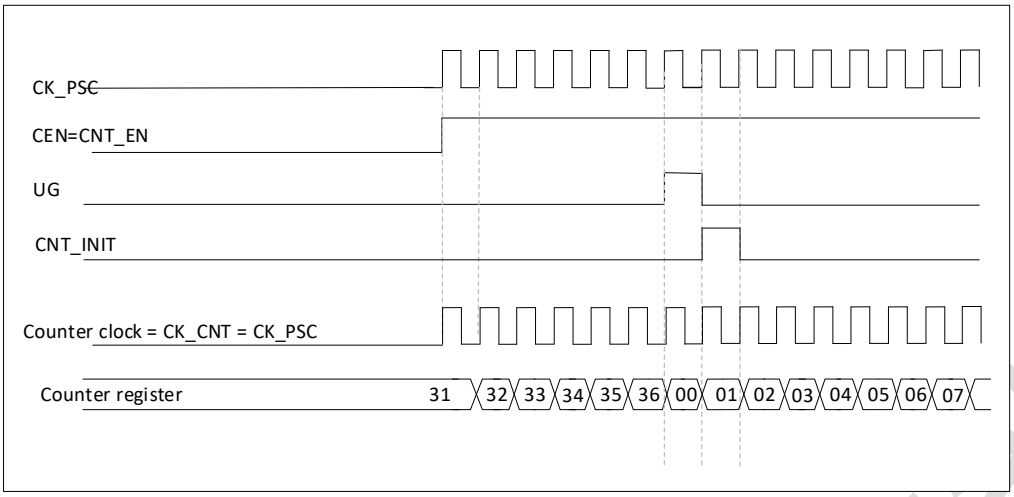


Figure 13-22 Control circuit in general mode with internal clock division factor 1

13.2.4.2. External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count on each rising or falling edge of the selected input.

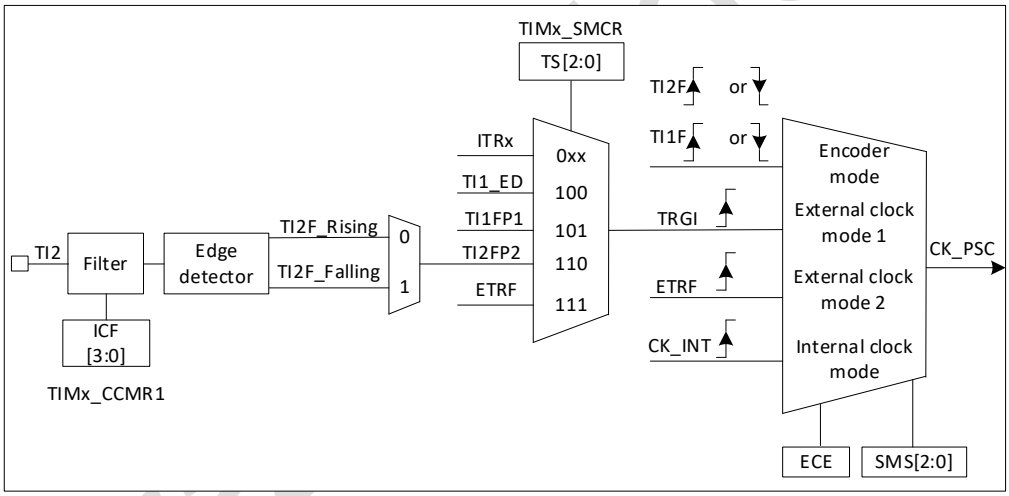


Figure 13-23 TI2 External Clock Connection Example

For example, to configure the counter to count up on the rising edge of the T12 input, use the following steps:

- Configure TIMx_CMR1 register CC2S=01 so that channel 2 detects the rising edge at the TI2 input;
- Configure IC2F[3:0] of the TIMx_CCMR1 register to select the input filter bandwidth (if no filter is required, keep IC2F=0000);
- Configure CC2P=0 of the TIMx_CCER register to select the rising edge polarity;
- Configure SMS=111 of the TIMx_SMCR register to select the timer for external clock mode 1;
- configure TS=110 in the TIMx_SMCR register to select TI2 as the trigger input source;
- Set CEN=1 in the TIMx_CR1 register to start the counter.

Note: The capture prescaler is not used as a trigger, so there is no need to configure it.

When the rising edge occurs at TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge at TI2 and the actual clock of the counter depends on the resynchronisation circuit at the TI2 input.

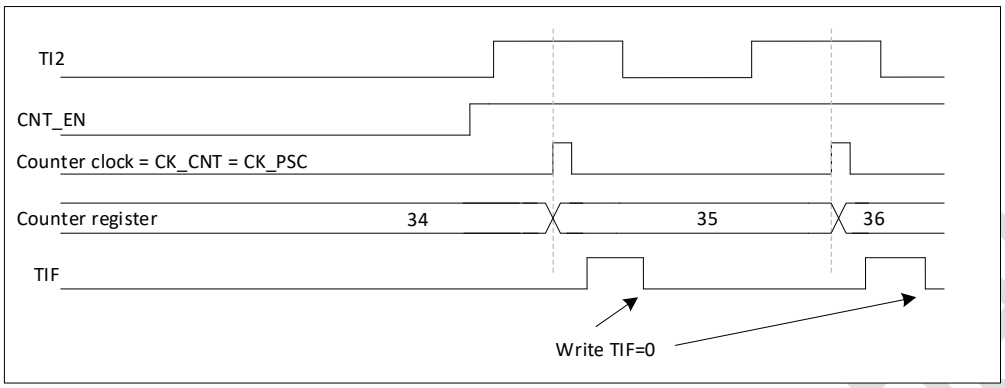


Figure 13-24 Control Circuit in External Clock Mode 1

13.2.4.3. External clock source mode 2

This mode is selected by making ECE=1 in the TIMx_SMCR register, and the counter is able to count on every rising or falling edge of the externally triggered ETR.

The following figure shows the block diagram of the externally triggered input:

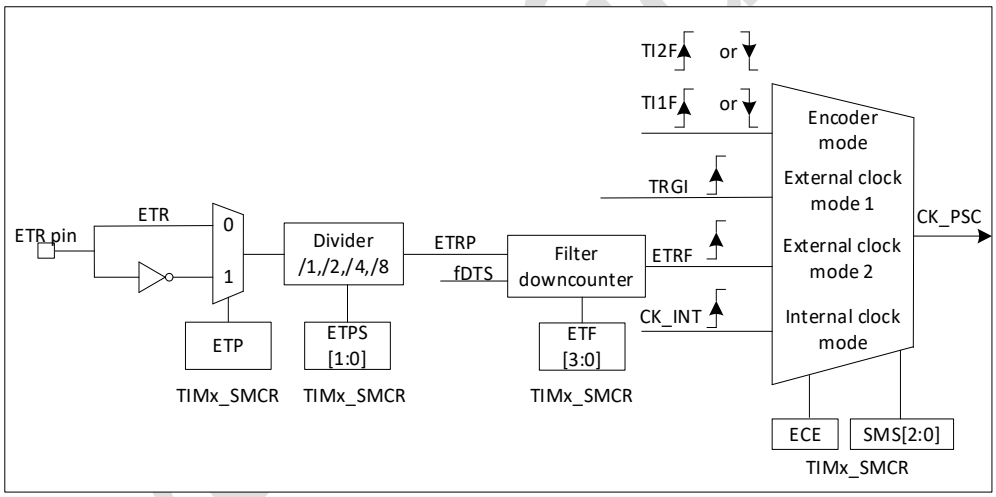


Figure 13-25 Block Diagram of External Trigger Inputs

For example, to configure an up counter that counts every 2 rising edges under ETR, use the following steps:

- No filter is needed in this example, set ETF[3:0]=0000 in the TIMx_SMCR register;
- Set the prescaler, set ETPS[1:0]=01 in the TIMx_SMCR register;
- Select the rising edge of ETR input, set ETP=0 in TIMx_SMCR register;
- Enable external clock mode 2, write ECE=1 in the TIMx_SMCR register;
- start the counter, write CEN=1 in the TIMx_CR1 register;

The counter counts every 2 rising edges of the ETR.

The delay between the rising edge of the ETR and the actual clock of the counter depends on the resynchronisation circuit of the ETRP signal.

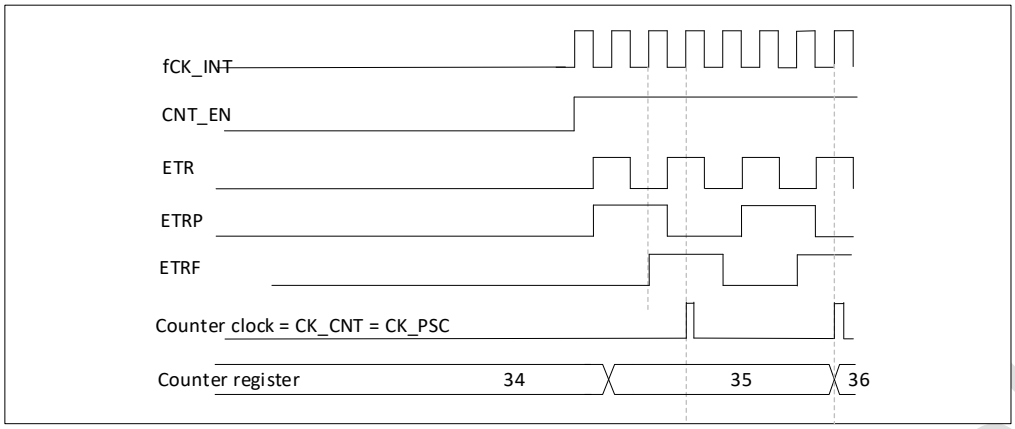


Figure 13-26 Control circuit in external clock mode 2

13.2.5. Capture/Compare Channel

Each capture/compare channel is centered around a capture/compare register (containing shadow registers), including the input portion of the capture (digital filtering, multiplexing and prescaler), and the output portion (comparator and output control).

The input section samples the corresponding TIx input signal and generates a filtered signal, TIxF. An edge monitor with polarity selection then generates a signal (TIxFPx) which can be used as an input trigger from the mode controller or as a capture control. This signal is pre-divided into a capture register (ICxPS). The ICxPS is obtained by dividing the frequency before the capture register.

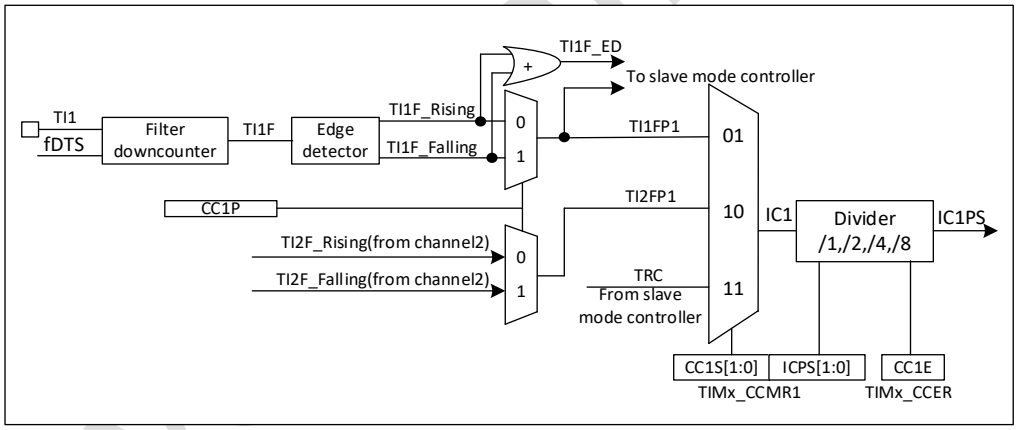


Figure 13-27 Capture/compare channels (e.g. Channel 1 input section)

The output section generates an intermediate waveform OCxRef (highly active) as a reference, and the end of the chain determines the polarity of the final output signal.

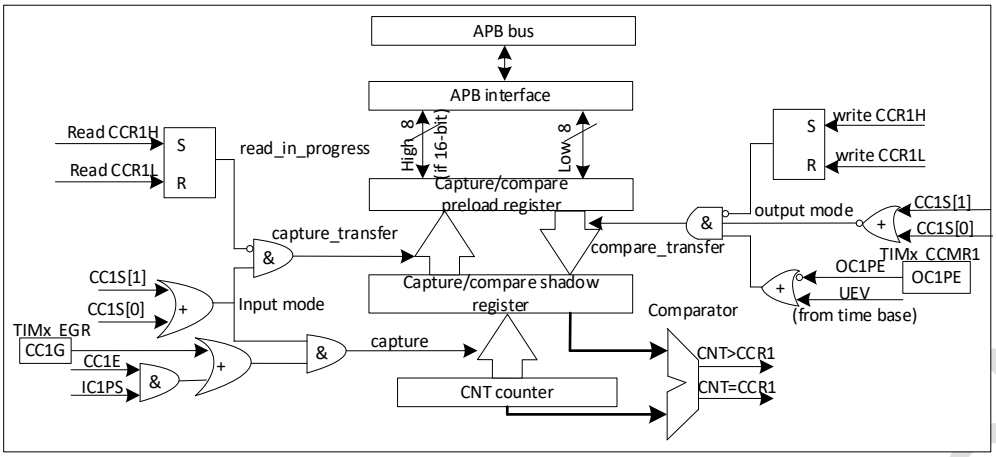


Figure 13-28 Main Circuit for Capture/Compare Channel 1

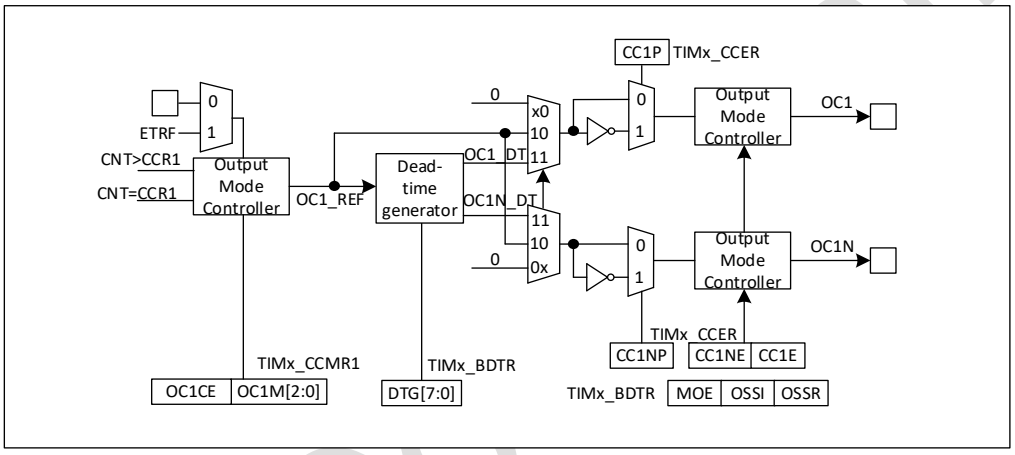


Figure 13-29 Output section of the capture/compare channel (channels 1 to 3)

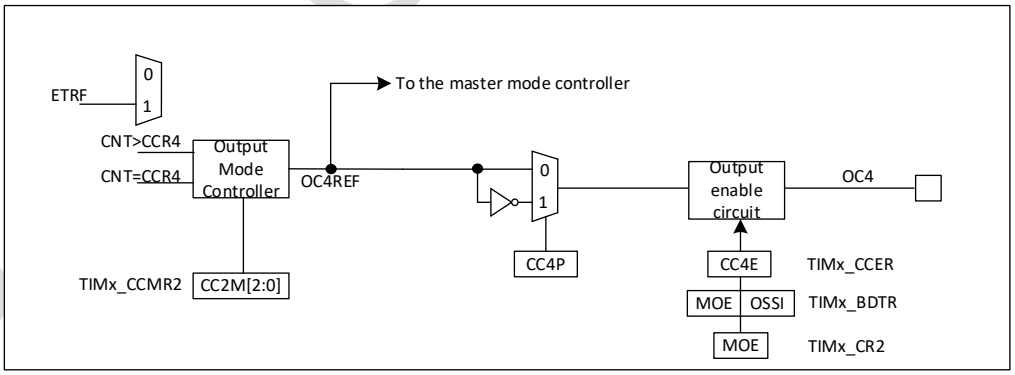


Figure 13-30 Output section of capture/compare channel (channel 4)

The capture/compare module consists of a preloaded register and a shadow register. The read/write process operates only on the preload register.

In capture mode, the capture occurs on the shadow register, which is then copied to the preload register.

In compare mode, the contents of the preload register are copied to the shadow register, and then the contents of the shadow register are compared to the counter.

13.2.6. Input capture mode

In input capture mode, the current value of the counter is latched into the capture/compare register (TIMx_CCRx) when the corresponding edge on the ICx signal is detected. When a capture event occurs, the corresponding CCxIF flag (TIMx_SR register) is set to 1, and an interrupt or DMA request will be generated if an interrupt or DMA operation is open. If the CCxIF flag is already high when a capture event occurs, the overcapture flag CCxOF (TIMx_SR register) is set to 1. CCxIF can be cleared by writing CCxIF=0, or by reading the capture data stored in the TIMx_CCRx register. writing CCxOF=0 clears CCxOF.

The following example shows how to capture the counter value into the TIMx_CMR1 register on the rising edge of the TI1 input as follows:

- Selecting Valid Inputs: TIMx_CMR1 must be connected to the TI1 input, so write CC1S=01 to the TIMx_CCR1 register, and as long as CC1S is not '00', the channel is configured as an input and the TIMx_CCR1 register becomes read-only.
- Configure the input filter for the desired bandwidth based on the characteristics of the input signal (i.e., when the input is TIx, the input filter control bit is the ICxF bit in the TIMx_CMRx register). Assuming that the input signal dithers over a period of up to 5 internal clock cycles, we have to configure the filter with a bandwidth longer than 5 clock cycles; we can therefore (at the fDTS frequency) sample the input 8 times in a row in order to confirm a true edge shift on TI1, i.e., by writing IC1F=0011 in the TIMx_CCMR1 register.
- Select the valid transition edge on the TI1 channel by writing CC1P=0 (set to rising edge) in the TIMx_CCER register.
- Configure the input prescaler. In this example, we want the capture to occur at every valid level transition moment, so the prescaler is disabled (write IC1PS=00 in the TIMx_CMR1 register).
- Set CC1E=1 of the TIMx_CCER register to allow the capture counter value to be captured into the capture register.
- If desired, allow related interrupt requests by setting the CC1IE bit in the TIMx_DIER register, and also allow DMA requests by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- A valid level transition is generated; the counter value is transferred to the TIMx_CCR1 register.
- The CC1IF flag bit is set (interrupt flag). When at least 2 consecutive captures occur and CC1IF has not been cleared, CC1OF is also set to 1.
- If the CC1IE bit is set, an interrupt is generated.
- If the CC1DE bit is set, a DMA request is also generated.

In order to handle overcatch, it is recommended to read the data before the overcatch flag is set, this is to avoid losing capture overflow information that may be generated after the capture overflow flag is read and before the data is read.

Note: Input capture interrupts and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

13.2.7. PWM Input Mode

This mode is a special case of the Input Capture mode and operates the same as the Input Capture mode except for the following differences:

- The two ICx signals are mapped to the same TIx input.
- The 2 ICx signals are edge-valid, but of opposite polarity.
- One of the TIxFP signals is used as the trigger input signal and the slave mode controller is configured in reset mode.

For example, the user can measure the period (TIMx_CCR1 register) and duty cycle (TIMx_CCR2 register) of the PWM signal input to TI1 as follows (depending on the frequency of CK_INT and the value of the prescaler).

- Select the valid input of TIMx_CCR1: set CC1S=01 of TIMx_CMR1 register (TI1 selected).
- Select valid polarity of TI1FP1 (used to capture data into TIMx_CCR1 and clear the counter): set CC1P=0 (valid on rising edge).
- Select valid input for TIMx_CCR2: set CC2S=10 of TIMx_CMR1 register (TI1 selected).
- Select valid polarity of TI1FP2 (capture data to TIMx_CCR2): set CC2P=1 (falling edge valid).
- Select valid trigger input signal: set TS=101 in TIMx_SMCR register (select TI1FP1).
- Configure slave mode controller to reset mode: set SMS=100 in TIMx_SMCR.
- Enable capture: set CC1E=1 and CC2E=1 in the TIMx_CCER register.

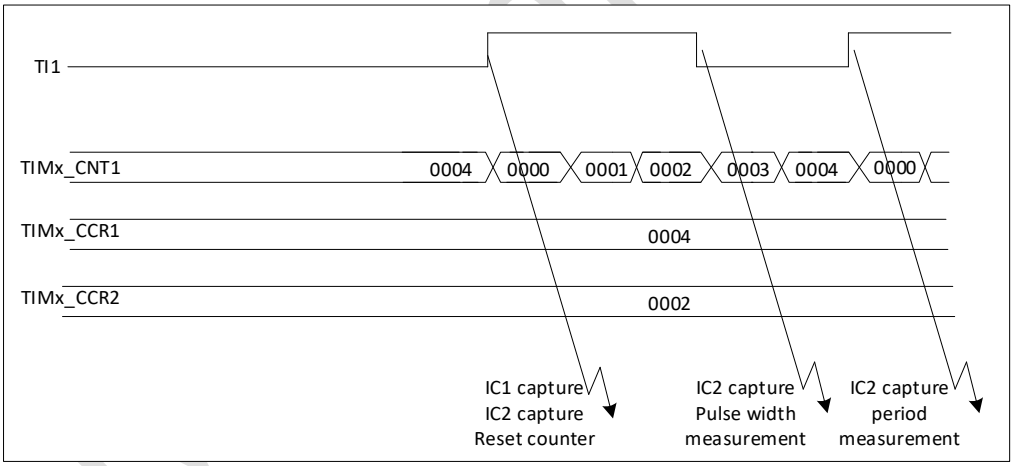


Figure 13-31 PWM Input Mode Timing

Since only TI1FP1 and TI2FP2 are connected to the Slave Mode Controller, only the TIMx_CH1/TIMx_CH2 signals can be used for PWM input mode.

13.2.8. Forced output mode

In output mode (CCxS=00 in TIMx_CCMRx register), the output compare signals (OCxREF and corresponding OCx/OCxN) can be forced to active or inactive state directly by the software independently of the result of the comparison between the output compare registers and counters.

The output compare signal (OCxREF/OCx) can be forced to a valid state by setting the corresponding OCxM=101 in the TIMx_CMRx register. In this way, OCxREF is forced high (OCxREF is always active high), while OCx gets the signal of CCxP with opposite polarity.

For example: CCxP=0 (OCx active high), then OCx is forced high.

Setting OCxM=100 in the TIMx_CMRx register forces the OCxREF signal low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter is still in progress and the corresponding flags are modified. Therefore, corresponding interrupts and DMA requests are still generated. This will be described in the Output Compare Mode section below.

13.2.9. Output comparison mode

This function is used to control an output waveform or to indicate that a given period of time has elapsed.

When the counter and the capture/compare registers have the same contents, the output compare function does the following:

- Outputs the values defined by the output compare mode (OCxM bit in the TIMx_CCMRx register) and output polarity (CCxP bit in the TIMx_CCER register) to the corresponding pins. The output pin can hold its level (OCxM=000), be set to an active level (OCxM=001), be set to an inactive level (OCxM=010), or be flipped (OCxM=011) when comparing matches.
- Set the flag bit in the interrupt status register (CCxIF bit in the TIMx_SR register).
- An interrupt is generated if the corresponding interrupt mask (CCxIE bit in the TIMx_DIER register) is set.
- A DMA request is generated if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register selects the DMA request function).

The need for the TIMx_CCRx registers to use preloaded registers can be selected by configuring the OCxPE bit in TIMx_CMRx.

In output compare mode, the update event UEV has no effect on the OCxREF and OCx outputs. The synchronisation can be done with an accuracy of up to one count cycle of the counter. The output compare mode (in single pulse mode) can also be used to output a single pulse.

Configuration steps for output compare mode:

- Select the counter clock (internal, external, prescaler).
- Write the appropriate data to the TIMx_ARR and TIMx_CCRx registers.
- Set the CCxIE bit if an interrupt request is to be generated.
- Selects the output mode, for example:
  - To request the counter to flip the OCx output pin when it matches CCRx, set OCxM = 011
  - Set OCxPE = 0 to disable preloaded registers
  - Set CCxP = 0 to select polarity active high
  - Set CCxE = 1 to enable outputs
- Set CEN bit of TIMx_CR1 register to start counter

The TIMx_CCRx registers can be updated by software at any time to control the output waveform, provided that the preloaded registers are not used (OCxPE='0', otherwise the shadow registers of TIMx_CCRx can only be updated when the next update event occurs). An example is given in the figure below.

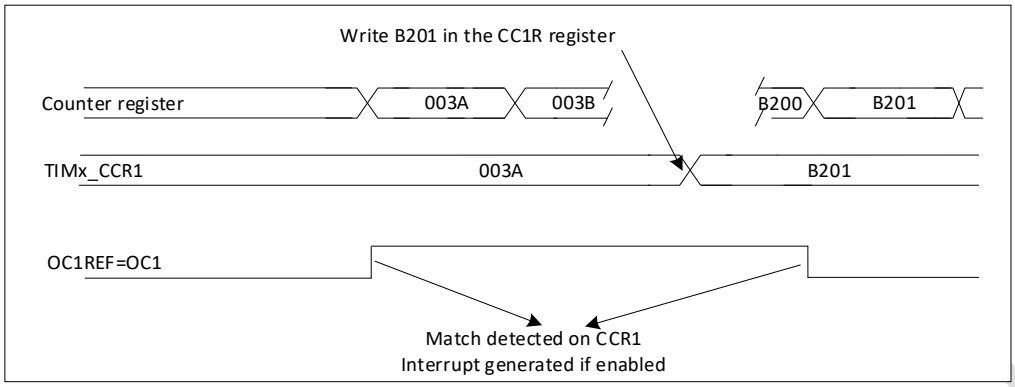


Figure 13-32 Output Compare Mode, Flip OC1

13.2.10. PWM Mode

Pulse width modulation mode can generate a signal with a frequency determined by the TIMx_ARR register and a duty cycle determined by the TIMx_CCRx register.

Writing '110' (PWM mode 1) or '111' (PWM mode 2) to the OCxM bit in the TIMx_CMRx register enables each OCx output channel to be set independently to generate a PWM all the way through. this must be done by setting the corresponding preload registers must be enabled by setting the OCxPE bit of the TIMx_CMRx register, and finally the ARPE bit of the TIMx_CR1 register, which (in up-count or centre-symmetric modes) enables the auto-reloaded preload registers.

The preloaded registers are transferred to the shadow registers only when an update event occurs, so the user must initialize all registers by setting the UG bit in the TIMx_EGR register before the counter starts counting.

The polarity of the OCx can be set by software with the CCxP bit in the TIMx_CCER register, which can be set to active high or active low. The output enable of the OCx is controlled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (in the TIMx_CCER and TIMx_BDTR registers).

In PWM mode (Mode 1 or Mode 2), TIMx_CNT and TIMx_CCRx are always being compared, (based on the counter count direction) to determine compliance with TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx.

Depending on the state of the CMS bit in the TIMx_CR1 register, the timer is able to generate either an edge-aligned PWM signal or a centre-aligned PWM signal.

13.2.10.1. PWM Edge Alignment Mode

- Upward Count Configuration

Up counting is performed when the DIR bit in the TIMx_CR1 register is low.

The following is an example for PWM mode 1. The PWM reference signal OCxREF is high when TIMx_CNT < TIMx_CCRx and low otherwise. If the comparison value in TIMx_CCRx is greater than the auto-reload value (TIMx_ARR), OCxREF remains '1'. If the comparison value is 0, OCxREF remains '0'. The following figure shows an example of an edge-aligned PWM waveform with TIMx_ARR=8.

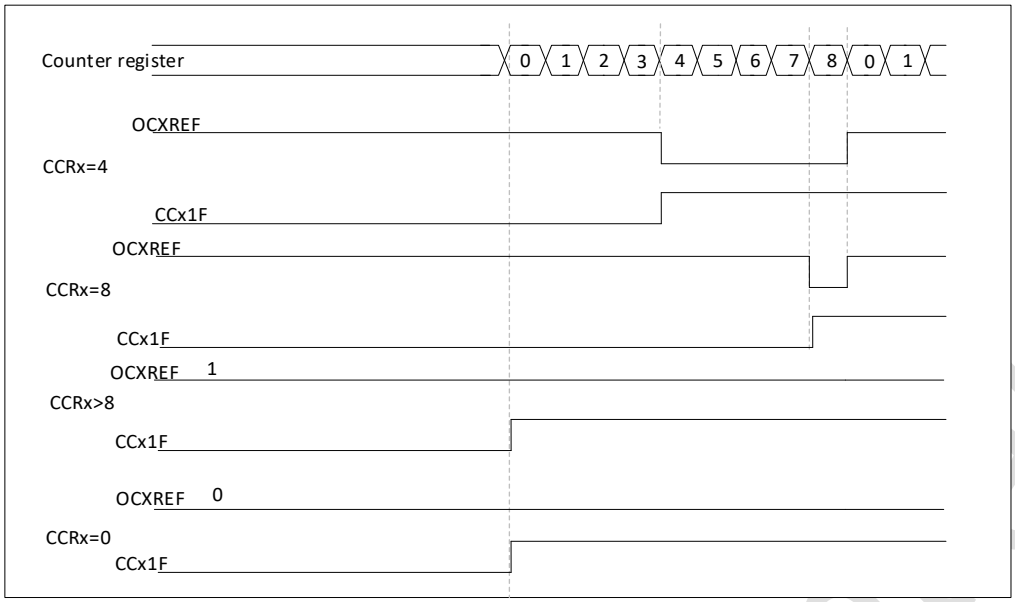


Figure 13-33 Edge-aligned PWM waveform (ARR=8)

- Configuration for Downward Counting

Downward counting is performed when the DIR bit of the TIMx_CR1 register is high.

In PWM mode 1, the reference signal OCxREF is low when TIMx_CNT > TIMx_CCRx and high otherwise. If the comparison value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, OCxREF remains '1'. A 0% PWM waveform cannot be generated in this mode.

13.2.10.2. PWM Central Alignment Mode

Central alignment mode when the CMS bit in the TIMx_CR1 register is not '00' (all other configurations of the CMS bit have the same effect on the OCxREF/OCx signals). Depending on the CMS bit setting, the compare flag can be set to 1 when the counter is counting up, 1 when the counter is counting down, or 1 when the counter is counting both up and down. The Count Direction Bit (DIR) in the TIMx_CR1 register is updated by hardware; do not modify it in software.

The following figure gives some examples of centre-aligned PWM waveforms

- TIMx_ARR=8
- PWM mode 1
- CMS=01 in TIMx_CR1 register sets the compare flag when the counter counts down in Central Alignment Mode 1.

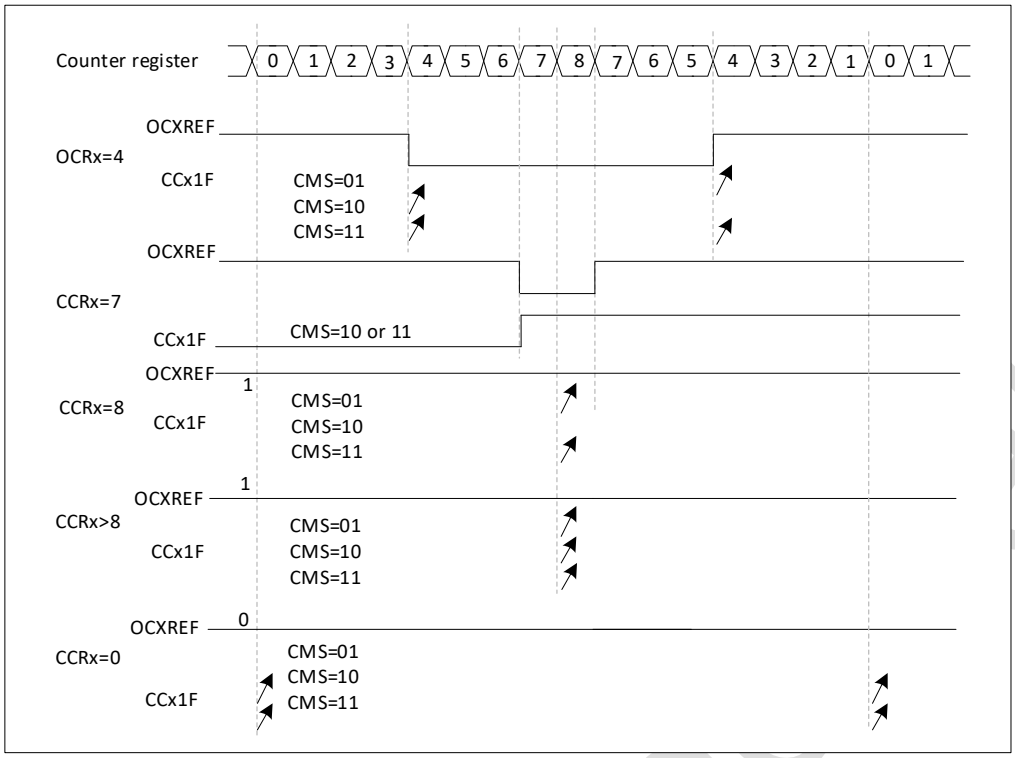


Figure 13-34 Centrally aligned PWM waveform (APR=8)

Hints for using Central Alignment Mode:

- The current count up/down configuration is used when entering central alignment mode; this means that whether the counter counts up or down depends on the current value of the DIR bit in the TIMx_CR1 register. Also, do not modify both the DIR and CMS bits via software.
- It is not recommended to rewrite the counter when running in centre-aligned mode, as this can produce unpredictable results. In particular:
  - If the value written to the counter is greater than the auto-reload value (TIMx_CNT>TIMx_ARR), the direction will not be updated. For example, if the counter is counting up, it continues to count up.
  - If a value of 0 or TIMx_ARR is written to the counter, the direction is updated, but no update event UEV is generated.
- The safest way to use the central alignment mode is to generate a software update (set the UG bit in the TIMx_EGR bit) before starting the counter and not to modify the counter value while the count is in progress.

13.2.11. Complementary outputs and deadband insertion

The advanced control timers (TIM1 and TIM8) are capable of outputting two complementary signals and manage the instantaneous switching off and on of the outputs.

This period of time is commonly referred to as the deadband, and the user should adjust the deadband time according to the connected output devices and their characteristics (delay of level transitions, delay of power switches, etc.).

Configuring the CCxP and CCxNP bits in the TIMx_CCER register allows you to select the polarity (main output OCx or complementary output OCxN) independently for each output.

The complementary signals OCx and OCxN are controlled by a combination of the following control bits: the CCxE and CCxNE bits of the TIMx_CCER register, the MOE, OISx, OISxN, OSSI, and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers, and the control bits for the complementary output channels OCx and OCxN with brakes. In particular, the deadband is activated on transition to the IDLE state (MOE falling to 0).

Setting both the CCxE and CCxNE bits will insert the deadband, and the MOE bit is also set if a brake circuit is present. The deadband generators for all channels can be controlled by configuring the DTG[7:0] bits in the TIMx_BDTR register. The reference signal OCxREF can generate 2 outputs OCx and OCxN. if OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal, except that its rising edge has a delay relative to the rising edge of the reference signal.
- The OCxN output signal is the opposite of the reference signal, except that its rising edge has a delay relative to the falling edge of the reference signal.

If the delay is greater than the currently valid output width (OCx or OCxN), the corresponding pulse is not generated.

The following graphs show the relationship between the output signal of the deadband generator and the current reference signal OCxREF. (Assuming CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1)

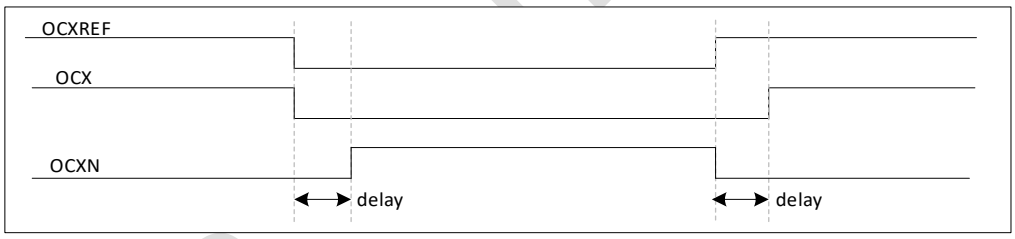


Figure 13-35 Complementary Outputs with Deadband Insertion

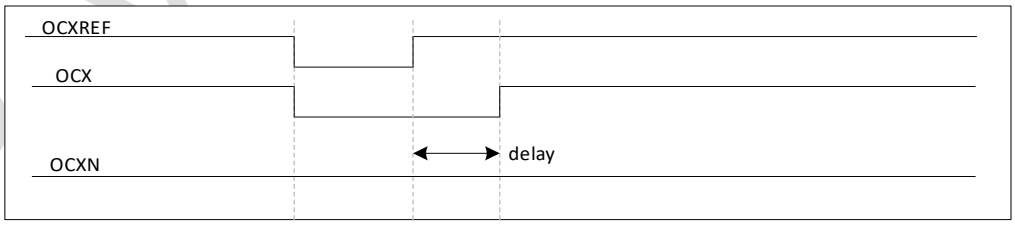


Figure 13-36 Deadband Waveform Delay Greater Than Negative Pulse

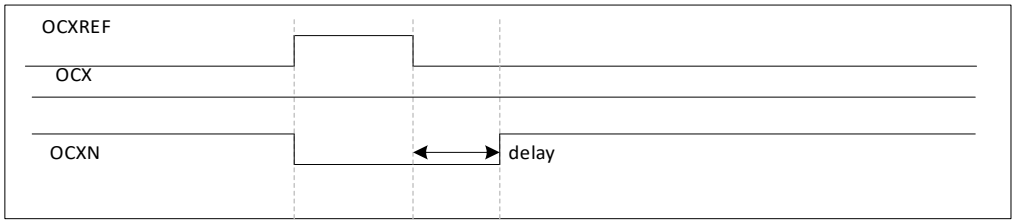


Figure 13-37 Deadband Waveform Delay Greater Than Positive Pulse

The deadband delay is the same for each channel and is configured programmatically by the DTG bit in the TIMx_BDTR register. See the specific delay calculations in the TIM1 and TIM8 brake and deadband registers (TIMx_BDTR) for details.

13.2.11.1. Redirection of OCxREF to OCx or OCxN

In output mode (strong set, output compare or PWM), OCxREF can be redirected to the output of OCx or OCxN by configuring the CCxE and CCxNE bits of the TIMx_CCER register.

This function allows a special waveform (e.g. PWM or static active level) to be sent on one of the outputs when the complementary output is at an invalid level. Another effect is to have both outputs at the same time at an invalid level, or at an active level and a complementary output with deadband.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not inverted and goes high immediately when OCxREF is active. For example, if CCxNP=0, then OCxN=OCxREF. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1), OCx is valid when OCxREF is high; and the opposite is true for OCxN, which becomes valid when OCxREF is low.

13.2.12. Use the brake function

When the brake function is used, based on the corresponding control bits (MOE, OSSI, and OSSR bits in the TIMx_BDTR register, and the OISx and OISxN bits in the TIMx_CR2 register), the output enable signals and invalid levels are modified. However, the OCx and OCxN outputs cannot be on active levels at the same time at the same time. See the control bits for the complementary output channels OCx and OCxN of the brake function for details.

The brake source can be either a brake input pin or a clock failure event. The clock fail event is generated by the clock safety system in the reset clock controller, see Clock Safety System (CSS) for details.

After a system reset, the brake circuit is disabled and the MOE bit is low. Setting the BKE bit in the TIMx_BDTR register enables the brake function, and the polarity of the brake input signal can be selected by configuring the BKP bit in the same register.BKE and BKP can be modified simultaneously. When writing the BKE and BKP bits, there is a delay of 1 APB clock cycle before the actual write, so it is necessary to wait for one APB clock cycle before the written bits can be read back correctly.

Because the MOE falling edge can be asynchronous, a resynchronisation circuit is set up between the actual signal (acting on the output) and the synchronisation control bit (in the TIMx_BDTR register). This resynchronisation circuit creates a delay between the asynchronous and synchronous signals. In particular, if MOE=1 is written when it is low, a delay (null instruction) must be inserted before it is read to get the correct value. This is because the write is asynchronous and the read is synchronous.

When a brake occurs (a selected level appears at the brake input), the following actions occur:

- The MOE bit is cleared asynchronously, placing the output in an invalid, idle, or reset state (selected by the OSSI bit). This feature remains in effect when the MCU's oscillator is turned off.

- Once MOE=0, each output channel outputs the level set by the OISx bit in the TIMx_CR2 register. If OSSI=0, the timer releases the enable output, otherwise the enable output is always high.
- When complementary outputs are used:
  - The output is first placed in a reset state i.e. an invalid state (depending on polarity). This is asynchronous operation and is valid even when the timer is not clocked.
  - If the clock of the timer is still present, the deadband generator will re-activate and drive the output port after the deadband according to the level indicated by the OISx and OISxN bits. Even in this case, OCx and OCxN cannot be driven to valid levels at the same time. Note that because of the resynchronisation of the MOE, the dead time is a bit longer than usual (about 2 ck_tim clock cycles).
  - The timer releases the enable output if OSSI=0, otherwise it holds the enable output; or the enable output goes high once one of CCxE and CCxNE goes high.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set at the next update event UEV; this can be used for shaping, for example. Otherwise, MOE always remains low until it is set '1' again; at this point, this feature can be used for safety, where you can connect the brake input to a power-driven alarm output, thermal sensor, or other safety device.

Note: The brake input is level active. Therefore, MOE cannot be set at the same time (automatically or via software) when the brake input is active. Also, the status flag BIF cannot be cleared.

When the brake is generated by the BRK input, its active polarity is programmable and is switched on by the BKE bit in the TIMx_BDTR register. The brake can also be generated by software setting the BG bit in the TIMx_EGR register.

In addition to the brake input and output management, write protection is implemented in the brake circuit to secure the application. It allows the user to freeze several configuration parameters (deadband duration, OCx/OCxN polarity and disabled state, OCxM configuration, brake enable and polarity). The user can select one of the three levels of protection by setting the LOCK bit in the TIMx_BDTR register, see TIM1 and TIM8 Brake and Deadband Registers (TIMx_BDTR). The LOCK bit can only be modified once after an MCU reset.

The following figure shows an example of the output in response to a brake.

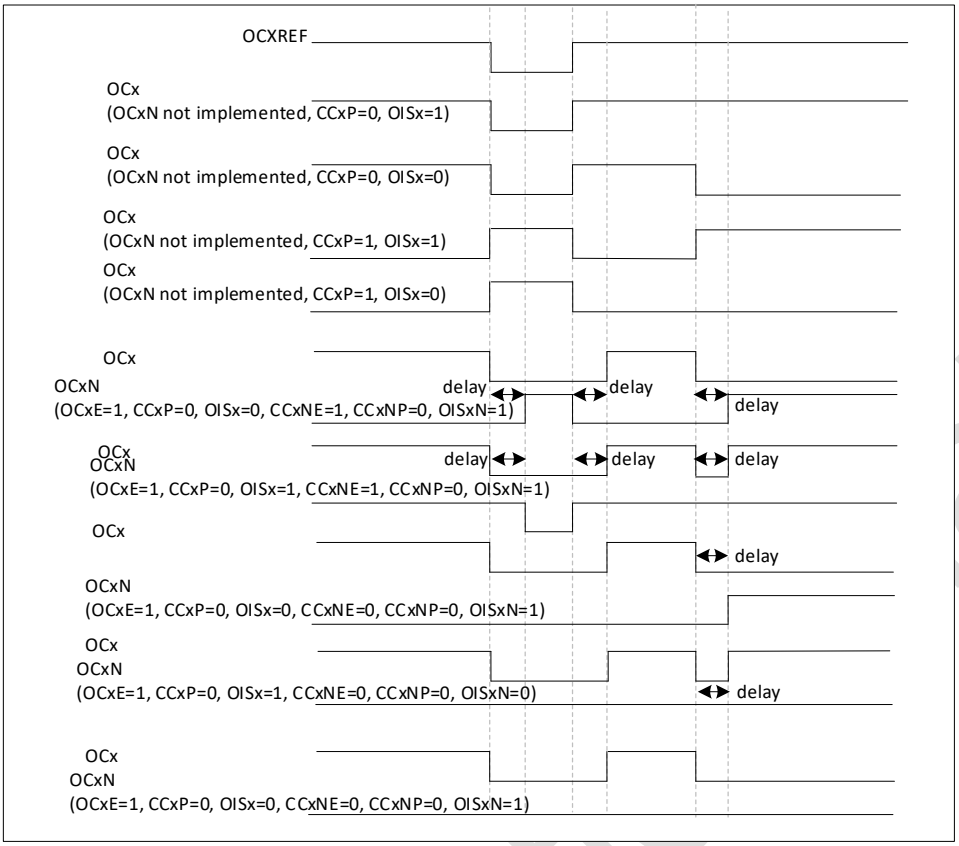


Figure 13-38 Outputs in response to braking

13.2.13. Clear the OCxREF signal on an external event

For a given channel, setting the corresponding OCxCE bit in the TIMx_CMRx register to '1' is able to pull the OCxREF signal low with a high level on the ETRF input, and the OCxREF signal will remain low until the next update event, UEV, occurs.

This function can only be used in output compare and PWM modes, not in force mode.

For example, the OCxREF signal can be coupled to the output of a comparator for current control. In this case, ETR must be configured as follows:

- The externally triggered prescaler must be off: ETPS[1:0]=00 in the TIMx_SMCR register.
- External clock mode 2 must be disabled: ECE=0 in the TIMx_SMCR register.
- The External Trigger Polarity (ETP) and External Trigger Filter (ETF) can be configured as required.

The following figure shows the action of the OCxREF signal when the ETRF input goes high, corresponding to different OCxCE values. In this example, the timer TIMx is placed in PWM mode.

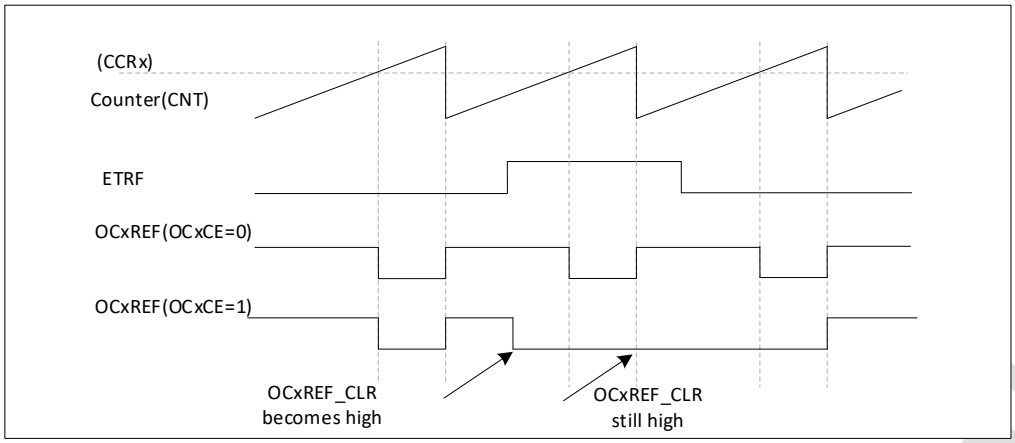


Figure 13-39 Clearing OCxREF for TIMx

13.2.14. Generate six-step PWM outputs

When complementary outputs are required on a channel, the preloaded bits are OCxM, CCxE and CCxNE. these preloaded bits are transferred to the shadow register bits when a COM phase change event occurs. This allows you to pre-set the next step configuration and fix the configuration of all channels at the same time.COM can be generated by software by setting the COM bit in the TIMx_EGR register, or by hardware at the rising edge of TRGI.

When a COM event occurs a flag bit (COMIF bit in the TIMx_SR register) is set, which generates an interrupt if the COMIE bit in the TIMx_DIER register has been set, or a DMA request if the COMDE bit in the TIMx_DIER register has been set.

The following figure shows the OCx and OCxN outputs for three different configurations when a COM event occurs.

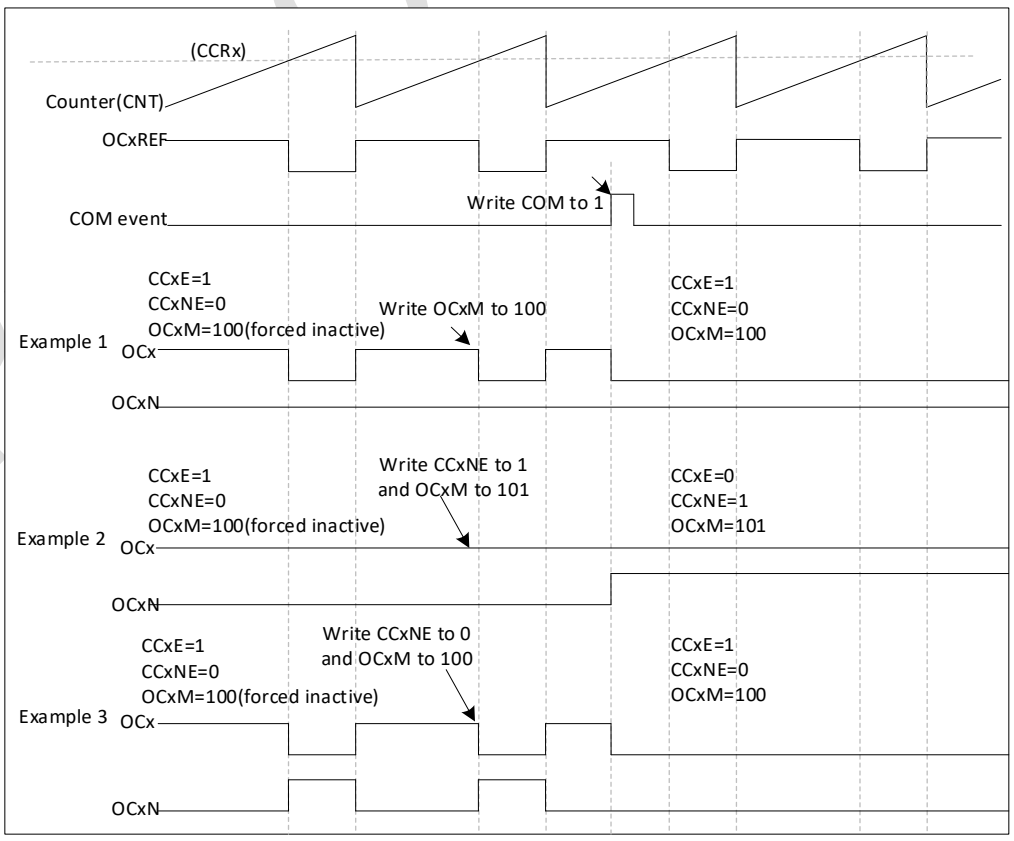


Figure 13-40 Example of generating a six-step PWM, using COM (OSSR=1)

13.2.15. Single-pulse mode

One-pulse mode (OPM) is a special case of one of the many modes described previously. This mode allows the counter to respond to an excitation and generate a pulse with a programmable pulse width after a programmable delay.

The counter can be activated from the mode controller to generate a waveform in either output comparison mode or PWM mode. Setting the OPM bit in the TIMx_CR1 register will select single pulse mode, which allows the counter to automatically stop when the next update event, UEV, is generated.

A pulse is generated only when the comparison value is different from the initial value of the counter. Before starting (when the timer is waiting to be triggered), it must be configured as follows:

- Upward counting mode: Counter CNT < CCRx ≤ ARR (in particular, 0 < CCRx).
- Downward counting mode: counter CNT > CCRx.

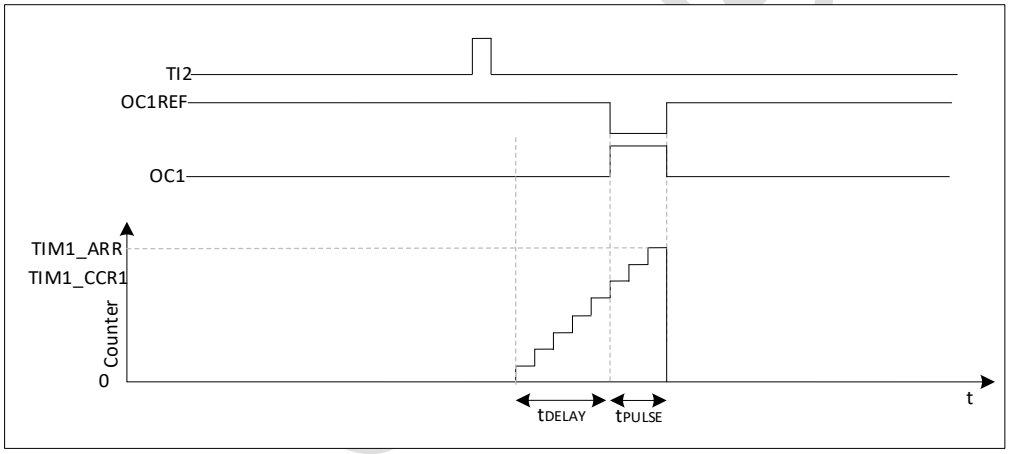


Figure 13-41 Example of single pulse mode

For example, you need to generate a positive pulse of length tPULSE on OC1 after a delay of tDELAY starting from the detection of a rising edge on the TI2 input pin.

Assuming that TI2FP2 acts as the trigger.

- Set CC2S=01 in the TIMx_CMR1 register to map TI2FP2 to TI2.
- Set CC2P=0 in the TIMx_CCER register to enable the TI2FP2 to detect rising edges.
- Set TS=110 in the TIMx_SMCR register, TI2FP2 triggers as a slave mode controller (TRGI).
- Set SMS=110 in the TIMx_SMCR register (trigger mode), the TI2FP2 is used to start the counter.

The OPM waveform is determined by the value written to the comparison register (taking into account the clock frequency and counter prescaler)

- tDELAY is defined by the value in the TIMx_CCR1 register.
- tPULSE is defined by the difference between the auto-load value and the compare value (TIMx_ARR - TIMx_CCR1).

- Assuming that a waveform from 0 to 1 is to be generated when a comparison match occurs, and a waveform from 1 to 0 is to be generated when the counter reaches the preload value; first set OC1M=111 in the TIMx_CMR1 register to enter PWM mode 2; selectively enable the preload registers as needed: set OC1PE=1 in TIMx_CMR1 and TIMx_CRM=1 in TIMx_CRM1 to enter PWM mode 2; set OC1PE=1 in TIMx_CMR1 and TIMx_CRM=1 in TIMx_CRM1 to enter PWM mode 2. 1 in TIMx_CCMR1 and ARPE in TIMx_CR1; then fill in the comparison value in TIMx_CCR1 register and the auto-load value in TIMx_ARR register, set the UG bit to generate an update event, and wait for an external trigger event on TI2. In this example, CC1P=0.
- In this example, the DIR and CMS bits in the TIMx_CR1 register should be set low.

Since only one pulse is required, OPM=1 in the TIMx_CR1 register must be set to stop counting at the next update event (when the counter flips from the auto-load value to 0). When OPM=0, repeat mode is selected.

13.2.15.1. Special case: OCx fast enable:

In single pulse mode, the CEN bit is set by the edge detection logic at the TIx input pin to start the counter. Comparison operations between the counter and the comparison value then produce the conversion of the output. However, these operations require a certain number of clock cycles, so it limits the minimum delay tDELAY that can be obtained.

To output a waveform with minimum delay, the OCxFE bit in the TIMx_CMRx register can be set; at this point OCxREF (and OCx) responds directly to the excitation and no longer relies on the result of the comparison, and the output waveform is the same as it would have been if the comparisons had been matched. OCxFE only works when the channel is configured for PWM1 and PWM2 modes.

13.2.16. Encoder interface mode

The encoder interface mode is selected by setting SMS=001 in the TIMx_SMCR register if the counter only counts on the TI2 edge, SMS=010 if it only counts on the TI1 edge, or SMS=011 if the counter counts on both the TI1 and TI2 edges.

TI1 and TI2 polarity can be selected by setting the CC1P and CC2P bits in the TIMx_CCER register; the input filters can also be programmed if required.

The two inputs TI1 and TI2 are used as interfaces to the incremental encoder. Assuming that the counter has been started (CEN=1 in the TIMx_CR1 register), the counter is driven by each valid trip on TI1FP1 or TI2FP2.TI1FP1 and TI2FP2 are the signals of TI1 and TI2 after passing them through the input filters and the polarity control; if there is no filtering and no phasing, then TI1FP1 = TI1 and TI2FP2 = The counting pulses and direction signals are generated according to the hopping sequence of the two input signals. Depending on the hopping sequence of the two input signals, the counter counts up or down while the hardware sets the DIR bit of the TIMx_CR1 register accordingly. Whether the counter counts on TI1, on TI2, or on both TI1 and TI2, a trip on either input (TI1 or TI2) recalculates the DIR bit.

The encoder interface mode is basically equivalent to using an external clock with direction selection. This means that the counter only counts continuously between 0 and the auto-loaded value of the TIMx_ARR register (either 0 to ARR counting or ARR to 0 counting, depending on the direction). So TIMx_ARR must be configured before counting starts; again, the capture, comparator,

prescaler, repeat counter, trigger output characteristics, etc. still work as usual. The encoder mode and the external clock mode 2 are not compatible and therefore cannot be operated at the same time. In this mode, the counter is automatically modified according to the speed and direction of the incremental encoder, so that the contents of the counter always indicate the position of the encoder. The counting direction corresponds to the direction of rotation of the connected sensor. The following table shows all possible combinations, assuming that TI1 and TI2 do not change at the same time.

Table 13-1 Count Direction vs. Encoder Signal

Effective Edge	Relative signal level (TI2 for TI1FP1, TI1 for TI2FP2))	TI1FP1Signal		TI2FP2Signal	
		Up	Down	Up	Down
Count on TI1 only	high	Count Down	Counting Up	Not counting	Not counting
	Low	Count Up	Count Down	No Count	Not Counting
Count on TI2 only	High	Not counting	Not counting	Count Up	Count Down
	Low	Not counting	Not counting	Count Down	Count Up
Count on TI1 and TI2	High	Count Down	Count Up	Count Up	Count Down
	Low	Count Up	Count Down	Count Down	Count Up

An external incremental encoder can be connected directly to the MCU without the need for external interface logic. However, a comparator is typically used to convert the differential output of the encoder to a digital signal, which greatly increases the immunity to noise interference. The third signal from the encoder output represents a mechanical zero, which can be connected to an external interrupt input and trigger a counter reset.

The figure below is an example of counter operation, showing the generation of the counting signal and direction control. It also shows how input jitter is suppressed when a double edge is selected; jitter may be generated when the sensor's position is close to a transition point. In this example, we assume the following configuration:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped to IC1)
- CC2S='01' (TIMx_CCMR2 register, TI2FP2 mapped to IC2)
- CC1P='0' (TIMx_CCER register, TI1FP1 not inverted, TI1FP1=TI1)
- CC2P='0' (TIMx_CCER register, TI2FP2 not inverted, TI2FP2=TI2)
- SMS='011' (TIMx_SMCR register, all inputs are valid on rising and falling edges).
- CEN='1' (TIMx_CR1 register, counter enable).

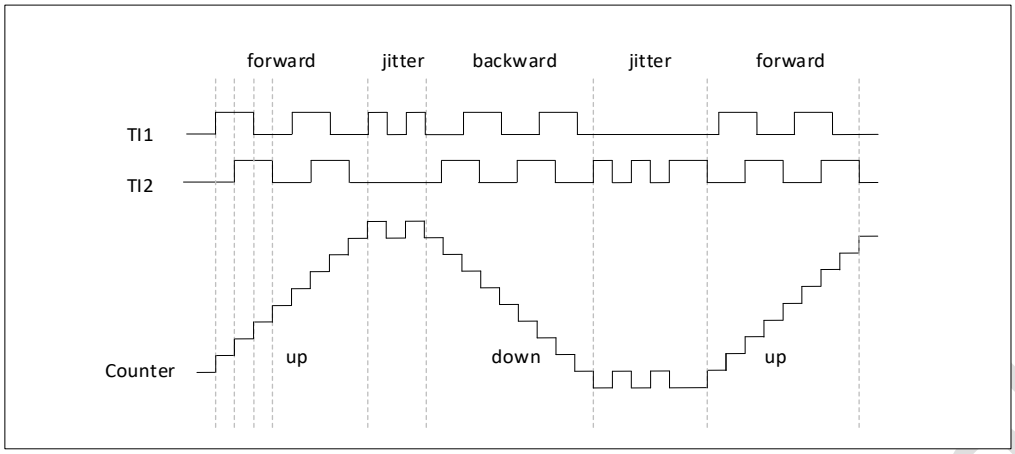


Figure 13-42 Example of Counter Operation in Encoder Mode

The following figure shows an example of counter operation when IC1FP1 polarity is inverted (CC1P='1', other configurations are the same as the above example)

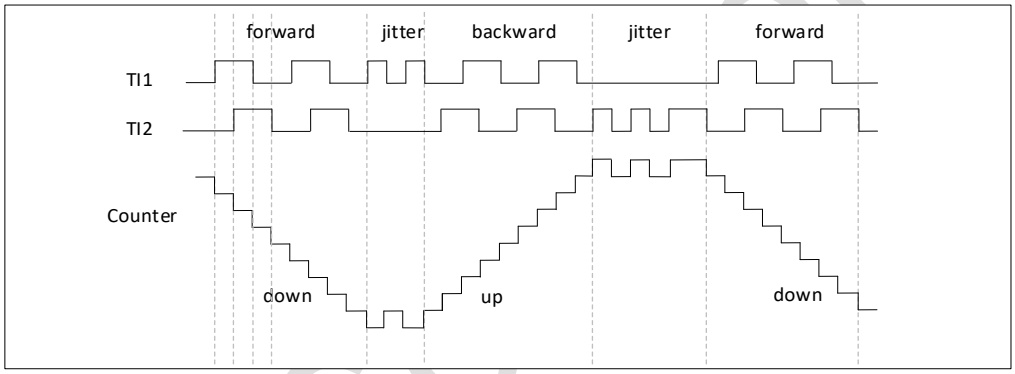


Figure 13-43 Example of encoder interface mode for IC1FP1 inversion

When the timer is configured in encoder interface mode, it provides information about the current position of the sensor. By configuring the second timer in capture mode, the interval between two encoder events can be measured to obtain information about the dynamics (velocity, acceleration, deceleration). The encoder output indicating the mechanical zero point can be used for this purpose. Depending on the interval between the two events, the counter can be read out at a fixed time. If possible, you can latch the value of the counter to a third input capture register (the capture signal must be periodic and can be generated by another timer); or you can read its value through a DMA request generated by the real-time clock.

13.2.17. Timer Input Isochronous Function

The TI1S bit in the TIMx_CR2 register allows the input filters of channel 1 to be connected to the outputs of a heterodyne gate, whose three inputs are TIMx_CH1, TIMx_CH2, and TIMx_CH3.

The heteros or outputs can be used for all timer input functions such as triggering or input capture. The following section gives an example of this feature being used to connect a Hall sensor.

13.2.18. Interface with Hall sensors

When using an advanced control timer (TIM1 or TIM8) to generate a PWM signal to drive the motor, another general-purpose TIMx (TIM2, TIM3, TIM4, or TIM5) timer can be used as an "interface timer" to connect to the Hall sensor, as shown in Figure 4-43, where the three timer input pins (See Figure 4-43. Three timer input pins (CC1, CC2, CC3) are connected to the TI1 input channel through an Iso-or-gate (selected by setting the TI1S bit in the TIMx_CR2 register), and the "interface timer" is used to capture this signal.

The slave mode controller is configured in reset mode, and the slave input is TI1F_ED. whenever one of the three inputs changes, the counter starts counting from 0 again. This produces a time reference triggered by any change in the Hall inputs.

"Capture/compare channel 1 on the Interface Timer is configured in capture mode and the capture signal is TRC (see Figure 4-26). The capture value reflects the time delay between two input changes and gives information about the motor speed.

The "interface timer" can be used to generate a pulse in the output mode, which can be used (by triggering a COM event) to change the attributes of each channel of the advanced timer TIM1 or TIM8, while the advanced control timer generates a PWM signal to drive the motor.

The "Interface Timer" channel must therefore be programmed to generate a positive pulse after a specified delay time (output comparison or PWM mode), which is sent to the Advanced Control Timer TIM1 or TIM8 via the TRGO output.

Example: A Hall input is connected to the TIMx timer, requiring the PWM configuration of the Advanced Control Timer TIMx to be changed at a specified moment after each change on either Hall input.

- Set the TI1S bit of the TIMx_CR2 register to '1' to configure the three timer inputs to be logically isochronous to the TI1 input;
- Time base programming: set TIMx_ARR to its maximum value (the counter must be cleared by a change in TI1). Set the prescaler to get a maximum counter period which is longer than the time interval between two changes on the sensor;
- Set channel 1 to capture mode (TRC checked): set CC1S=01 in the TIMx_CMR1 register, and set the digital filter if required;
- Set channel 2 to PWM2 mode with the requested delay: set OC2M=111 and CC2S=00 in the TIMx_CCMR1 register;
- Select OC2REF as the trigger output on TRGO: set MMS=101 in the TIMx_CR2 register.

In the advanced control register TIM1, the correct ITR input must be the trigger input, the timer is programmed to generate the PWM signal, the capture/compare control signal is pre-loaded (CCPC=1 in the TIMx_CR2 register), and the trigger input controls the COM event (CCUS=1 in the TIMx_CR2 register). After a COM event, the next PWM control bits (CCxE, OCxM) are written, which can be implemented in the interrupt subroutine that handles the rising edge of OC2REF.

The following figure shows this example

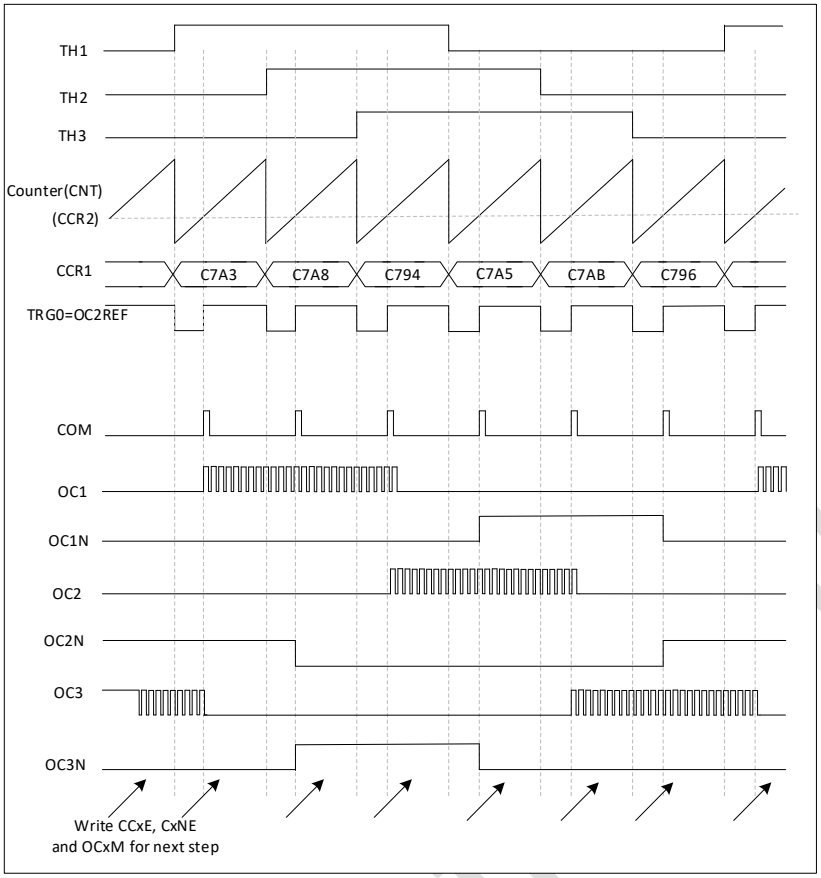


Figure 13-44 Example of Hall sensor interface

13.2.19. Synchronisation of TIMx timers and external triggers

The TIMx timer can be synchronised with an external trigger in several modes: reset mode, gated mode and triggered mode.

13.2.19.1. Slave mode: reset mode

On the occurrence of a triggered input event, the counter and its prescaler can be reinitialised; at the same time, an update event UEV is also generated if the UDIS bit of the TIMx_CR1 register is low; all preloaded registers (TIMx_ARR, TIMx_CCRx) are then updated.

In the following example, a rising edge on the TI1 input causes the up counter to be cleared to zero:

- Configure channel 1 to detect the rising edge of TI1. Configure the bandwidth of the input filter (in this example, no filter is required, so keep IC1F=0000). The capture prescaler is not used in the trigger operation, so no configuration is required. the CC1S bit selects the input capture source only, i.e., CC1S=01 in the TIMx_CMR1 register. set CC1P=0 in the TIMx_CCER register to determine the polarity (detects the rising edge only).
- Set SMS=100 in the TIMx_SMCR register to configure the timer for reset mode; set TS=101 in the TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in TIMx_CR1 register to start the counter.

The counter starts counting according to the internal clock, and then operates normally until a rising edge appears in TI1; at this time, the counter is cleared to zero and then restarts counting from zero. At the same time, the trigger flag (TIF bit in the TIMx_SR register) is set, generating either an

interrupt request or a DMA request depending on the setting of the TIE (interrupt enable) bit and the TDE (DMA enable) bit in the TIMx_DIER register.

The following figure shows the action when the auto-reload register TIMx_ARR = 0x36. The delay between the rising edge of TI1 and the actual reset of the counter depends on the resynchronisation circuitry at the TI1 input.

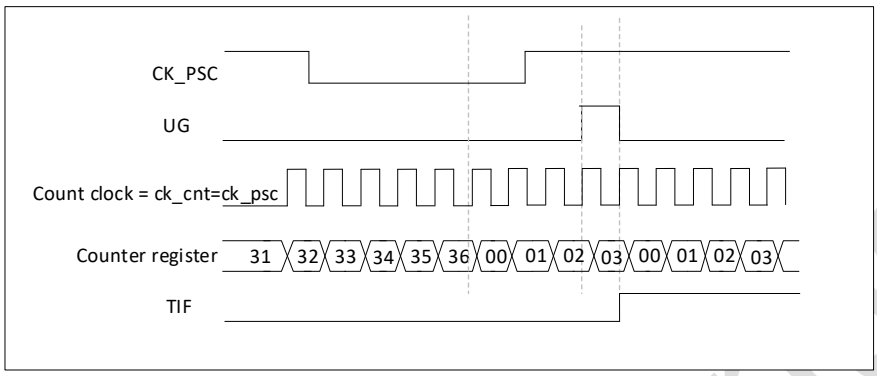


Figure 13-45 Control Circuit in Reset Mode

13.2.19.2. Slave mode: gated mode

Enable the counter according to the level of the selected input.

In the following example, the counter only counts up when TI1 is low:

- Configure channel 1 to detect a low level on TI1. Configure the input filter bandwidth (in this example, no filtering is required, so keep IC1F=0000). The capture prescaler is not used in the trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source, set CC1S=01 in the TIMx_CMR1 register. set CC1P=1 in the TIMx_CCER register to determine the polarity (detects only low levels).
- Set SMS=101 in the TIMx_SMCR register to configure the timer for gated mode; set TS=101 in the TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in TIMx_CR1 register to start the counter. In gated mode, if CEN=0, the counter cannot start, regardless of the trigger input level.

As long as TI1 is low, the counter starts counting based on the internal clock and stops counting once TI1 goes high. The TIF marker in TIMx_SR is set when the counter starts or stops.

The delay between the rising edge of TI1 and the actual stopping of the counter depends on the resynchronisation circuit at the TI1 input.

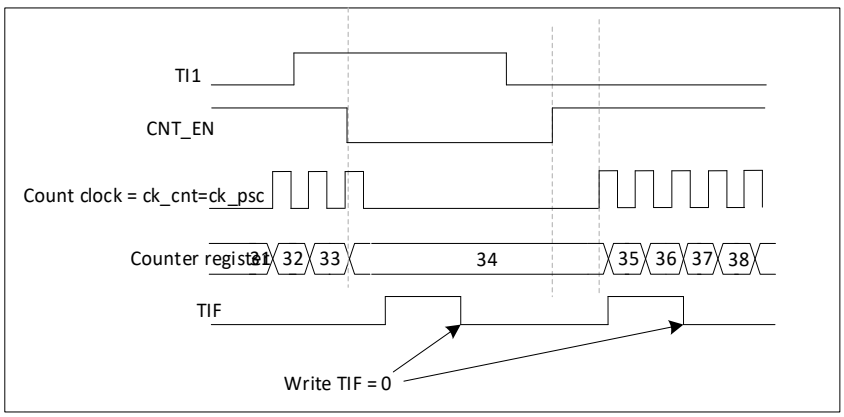


Figure 13-46 Control Circuit in Gated Mode

13.2.19.3. Slave mode: trigger mode

The selected event on the input enables the counter.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

- Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is required, keeping IC2F=0000). The capture prescaler is not used in the trigger operation and does not need to be configured. the CC2S bit is only used to select the input capture source, set CC2S=01 in the TIMx_CMR1 register. set CC2P=1 in the TIMx_CCER register to determine the polarity (detects only low levels).
- Set SMS=110 in the TIMx_SMCR register to configure the timer for trigger mode; set TS=110 in the TIMx_SMCR register to select TI2 as the input source.

When a rising edge of TI2 occurs, the counter starts counting driven by the internal clock and the TIF flag is set at the same time.

The delay between the rising edge of TI2 and the counter starting to count depends on the resynchronisation circuit at the TI2 input.

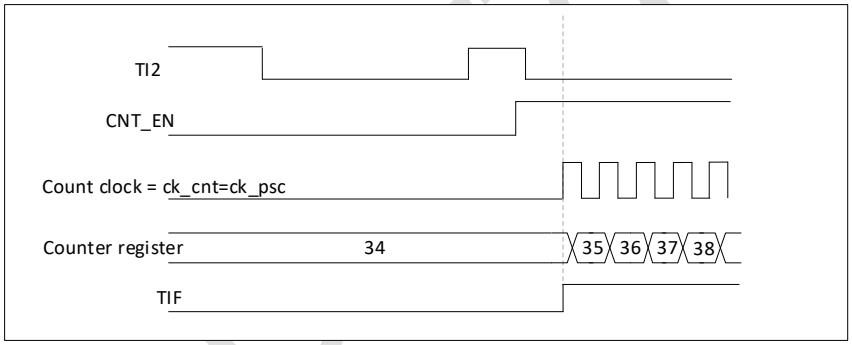


Figure 13-47 Control Circuit in Trigger Mode

13.2.19.4. Slave mode: external clock mode 2 + trigger mode

External clock mode 2 can be used with another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as an input to the external clock, and another input can be selected as a trigger input in reset mode, gated mode or trigger mode. It is not recommended to use the TS bit of the TIMx_SMCR register to select ETR as the TRGI.

In the following example, once a rising edge occurs on TI1, the counter counts up once on each rising edge of ETR:

- Configure the external trigger input circuit via the TIMx_SMCR register:
  - ETF=0000: no filtering
  - ETPS=00: without prescaler
  - ETP=0: detect the rising edge of ETR, set ECE=1 to enable external clock mode 2.
- Configure channel 1 as follows to detect the rising edge of TI:
  - IC1F=0000: no filtering

- Capture prescaler is not used in trigger operation, no configuration required
- Set CC1S=01 in the TIMx_CMR1 register to select the input capture source
- Set CC1P=0 in TIMx_CCER register to determine polarity (detects rising edge only)
- Set SMS=110 in the TIMx_SMCR register to configure the timer for trigger mode. Set TS=101 in the TIMx_SMCR register to select TI1 as the input source.

When a rising edge occurs on TI1, the TIF flag is set and the counter begins counting on the rising edge of ETR.

The delay between the rising edge of the ETR signal and the actual reset of the counter depends on the resynchronisation circuit at the ETRP input.

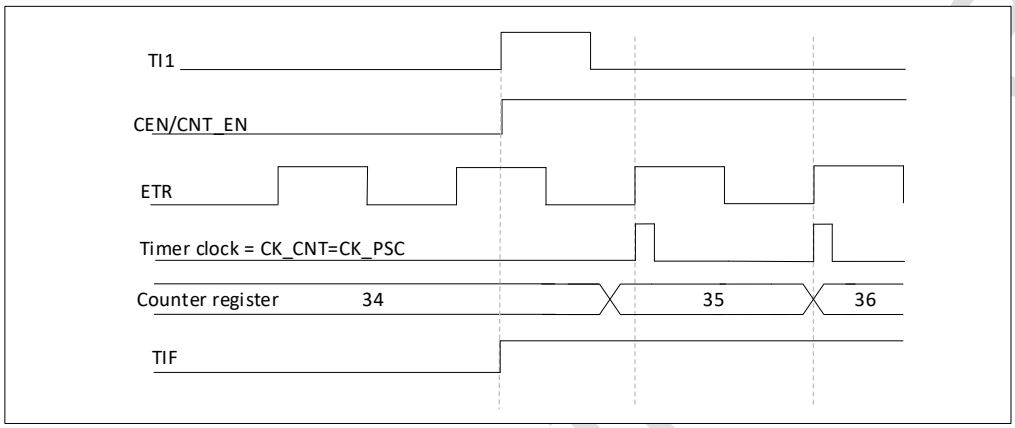


Figure 13-48 Control Circuit in External Clock Mode 2 + Trigger Mode

13.2.20. Timer Synchronisation

All TIMx timers are connected internally for timer synchronisation or linking. When one timer is in master mode, it can reset, start, stop, or provide a clock to the counter of another timer that is in slave mode.

The following figure shows an overview of the Trigger Select and Master Mode Select modules.

13.2.20.1. Use a timer as a prescaler for another timer

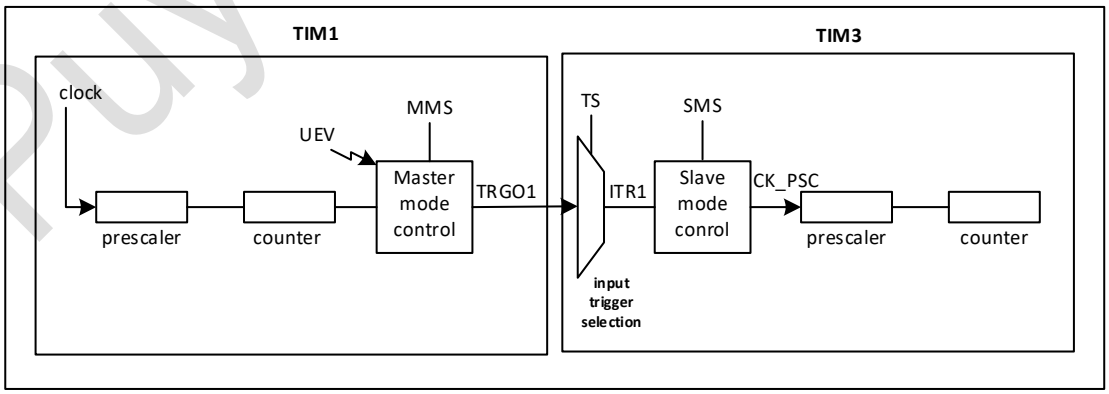


Figure 13-49 Example of a Master/Slave Timer

For example, you can configure Timer 1 as a prescaler for Timer 2. Perform the following operation:

- Configure Timer 1 as the master mode, which can output a periodic trigger signal at every update event UEV. With MMS='010' in the TIM1_CR2 register, output a rising edge signal on TRGO1 whenever an update event is generated.
- Connect the TRGO1 output of Timer1 to Timer2, set TS='000' of the TIM2_SMCR register, and configure Timer2 for slave mode using ITR1 as the internal trigger.
- The slave mode controller is then placed in external clock mode 1 (SMS=111 of the TIM2_SMCR register); this allows Timer 2 to be driven by the periodic rising edge (i.e. Timer 1's counter overflow) signal from Timer 1.
- Finally, the CEN bit of the corresponding (TIMx_CR1 register) must be set to start each of the two timers.

Note: If OCx has been selected as the trigger output of Timer 1 (MMS=1xx), its rising edge is used to drive the counter of Timer 2.

13.2.20.2. Use a timer to enable another timer

In this example, the enable of Timer 2 is controlled by the output comparison of Timer 1. Timer 2 counts the divided internal clock only when OC1REF of Timer 1 is high. The clock frequency of both timers is obtained by dividing CK_INT by 3 (fCK_CNT = fCK_INT/3) by the prescaler.

- Configure Timer1 as master mode, send its output compare reference signal (OC1REF) as trigger output (MMS=100 in TIM1_CR2 register)
- Configure Timer 1's OC1REF waveform (TIM1_CCMR1 register)
- Configure Timer 2 to get input trigger from Timer 1 (TS=000 for TIM2_SMCR register)
- Configure Timer 2 for gated mode (SMS=101 of TIM2_SMCR register)
- Set CEN=1 of TIM2_CR1 register to enable Timer 2
- Set CEN=1 of TIM1_CR1 register to enable Timer1

Note: The clock of Timer2 is not synchronised with the clock of Timer1, this mode only affects the enable signal of the Timer2 counter.

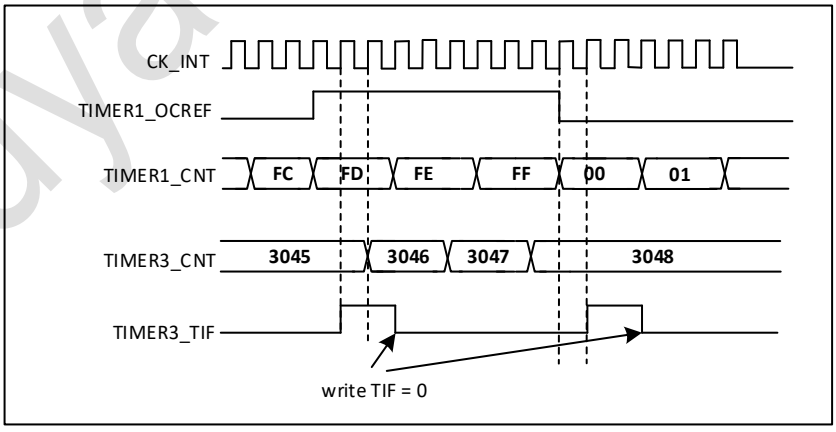


Figure 13-50 OC1REF for Timer 1 Controls Timer 2

In the example in Figure 4-47, before Timer 2 is started, their counters and prescalers are not initialised, so they start counting from the current value. It is possible to reset the 2 timers before

starting Timer 1 so that they start from a given value, i.e., write any value needed in the timer counter. The timers can be reset by writing the UG bit of the TIMx_EGR register.

In the next example, it is necessary to synchronise Timer 1 and Timer 2. Timer 1 is in master mode and starts from 0, Timer 2 is in slave mode and starts from 0xE7; the prescaler coefficients are the same for both timers. Writing '0' to the CEN bit of TIM1_CR1 will disable Timer 1 and Timer 2 will then stop.

- Configure Timer 1 as master mode, send output compare 1 reference signal (OC1REF) as trigger output (MMS=100 in TIM1_CR2 register).
- Configure the OC1REF waveform for Timer 1 (TIM1_CCMR1 register).
- Configure Timer2 to get input trigger from Timer1 (TS=000 for TIM2_SMCR register)
- Configure Timer 2 for gated mode (SMS=101 of TIM2_SMCR register)
- Set UG='1' of TIM1_EGR register to reset Timer 1.
- Set UG='1' of TIM2_EGR register to reset timer 2.
- Write '0XE7' to Timer 2's counter (TIM2_CNT) to initialise it to 0xE7.
- Set CEN='1' of TIM2_CR1 register to enable Timer 2.
- Set CEN='1' of TIM1_CR1 register to enable Timer 1.
- Set CEN='0' of TIM1_CR1 register to stop Timer 1.

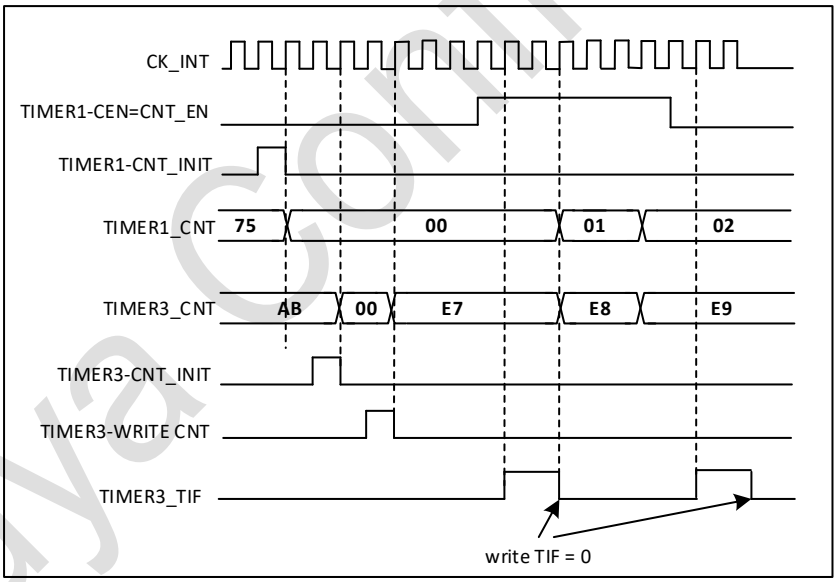


Figure 13-51 Timer 2 can be controlled by enabling Timer 1

13.2.20.3. Use a timer to start another timer

In this example, Timer 2 is enabled using an update event from Timer 1. As soon as Timer 1 generates an update event, Timer 2 starts counting from its current value (which can be non-zero) according to the divided internal clock. On receipt of a trigger signal, the CEN bit of Timer 2 is automatically set to '1' and the counter starts counting until a '0' is written to the CEN bit of the TIM2_CR1 register. Both timers are clocked by dividing the prescaler pair CK_INT by 3 (fCK_CNT = fCK_INT/3).

- Configure Timer 1 to be in master mode, sending its update event (UEV) as a trigger output (MMS=010 in the TIM1_CR2 register).
- Configure the period of Timer 1 (TIM1_ARR register).
- Configure Timer2 to get input trigger from Timer1 (TS=000 for TIM2_SMCR register)
- Configure Timer 2 for trigger mode (SMS=110 of TIM2_SMCR register)
- Set CEN=1 of TIM1_CR1 register to start Timer 1.

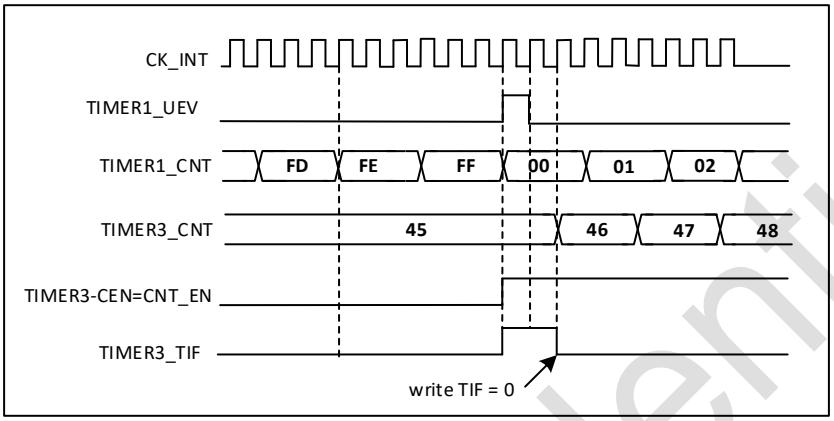


Figure 13-52 Triggering Timer 2 Using Timer 1 Updates

In the previous example, it is possible to initialise both counters before starting the count. Shows the action in the same configuration as 0, using trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

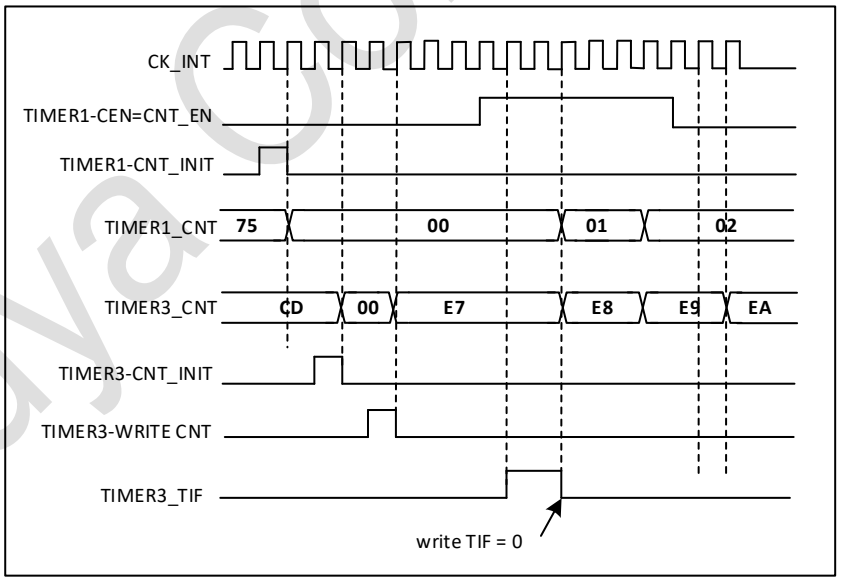


Figure 13-53 Triggering Timer 2 with Timer 1 Enable

13.2.20.4. Start 2 timers synchronously with an external trigger

This example enables Timer 1 when the TI1 input to Timer 1 rises, and enables Timer 1 while enabling Timer 2, see Figure 4-47. to ensure counter alignment, Timer 1 must be configured in master/slave mode (corresponding to TI1 as a slave and Timer 2 as a master):

- Configure Timer1 as master mode, send its enable as trigger output (MMS=001 in TIM1_CR2 register).
- Configure Timer1 as slave mode, get input trigger from TI1 (TS=100 of TIM1_SMCR register).
- Configure Timer1 for trigger mode (SMS=110 of TIM1_SMCR register).
- Configure Timer 1 for master/slave mode with MSM=1 of the TIM1_SMCR register.
- Configure Timer2 to get input trigger from Timer1 (TS=000 of TIM2_SMCR register)
- Configure Timer 2 for trigger mode (SMS=110 of TIM2_SMCR register).

When a rising edge occurs on TI1 of Timer 1, both timers synchronously start counting according to the internal clock and both TIF flags are set at the same time.

Note: In this example, both timers are initialised (set the corresponding UG bit) before start-up and both counters start from 0, but an offset can be inserted between the timers by writing to any of the counter registers (TIMx_CNT). In the figure below you can see that in master/slave mode there is a delay between CNT_EN and CK_PSC in timer 1.

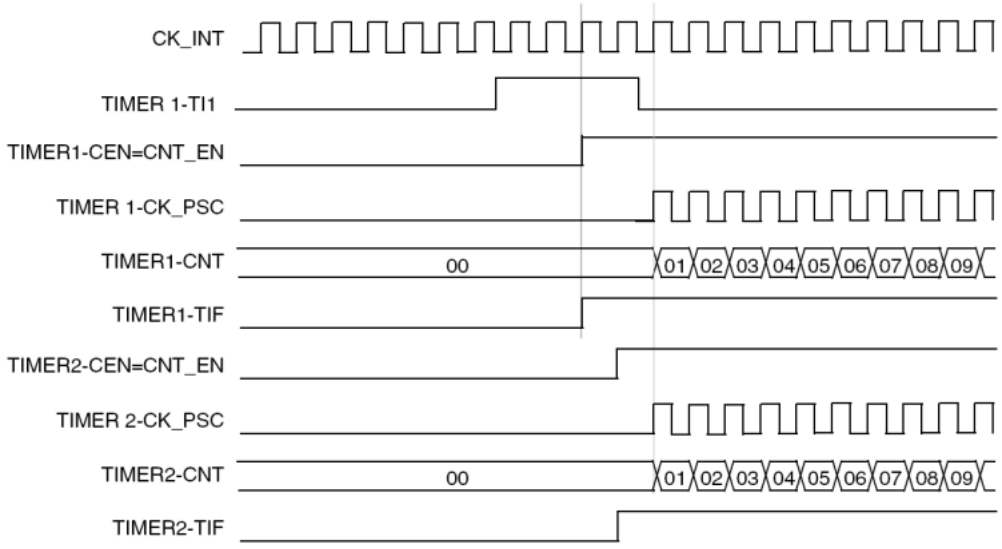


Figure 13-54 Triggering Timer 1 and Timer 2 Using the TI1 Input of Timer 1

13.2.21. Debug Mode

When the microcontroller enters debug mode (Cortex-M4 core stop), depending on the DBG_TIMx_STOP setting in the DBG module, the

TIMx counter can either continue normal operation or stop.

13.3. Register Descriptions

TIM1 register base address: 0x4001 2C00

TIM8 register base address: 0x4001 3400

13.3.1. TIM1 and TIM8 control registers1 (TIMx_CR1)

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
Reserved						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:10	Reserved, always 0			
9:8	CKD	RW	0	Clock division These two bits define the ratio between the timer clock (CK_INT) frequency, the dead time and the division between the dead time generator and the sampling clock (tDTS) used by the digital filter (ETR,TIx). 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: Reserved, do not use this configuration
7	ARPE	RW	0	Auto-reload preload enable bit 0: TIMx_ARR register is not buffered; 1: TIMx_ARR register is buffered.
6:5	CMS	RW	0	Center-aligned mode selection 00: Edge-aligned mode. The counter counts up or down according to the direction bit (DIR). 01: Centre-aligned mode 1. the counter alternately counts up and down. The output compare interrupt flag bit of the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set only when the counter is counting down. 10: Central Alignment Mode 2. the counter alternately counts up and down. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set only when the counter counts up. 11: Central Alignment Mode 3. the counter alternately counts up and down. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set both when the counter counts up and down. Note: While the counter is on (CEN=1), the transition from edge-aligned mode to centre-aligned mode is not allowed.
4	DIR	RW	0	Direction 0: Counter counts up; 1: Counter counts down. Note: This bit is read-only when the counter is configured for centre-aligned mode or encoder mode.
3	OPM	RW	0	One pulse mode 0: The counter does not stop when an update event occurs; 1: The counter stops when the next update event occurs (CEN bit is cleared).

Bit	Name	R/W	Reset Value	Function
2	URS	RW	0	Update request source Software selects the source of UEV events with this bit. 0: If an update interrupt or DMA request is enabled, either of the following events generates an update interrupt or DMA request: - Counter overflow/underflow - Setting of the UG bit - Update generated from the mode controller 1: If an update interrupt or DMA request is enabled, only a counter overflow/underflow generates an update interrupt or DMA request.
1	UDIS	RW	0	Update disable The software allows/disables the generation of UEV events with this bit 0: UEV is allowed. update (UEV) events are generated by any of the following events: - Counter overflow/underflow - Setting of the UG bit - Update generated from the mode controller Registers with caches are loaded with their preloaded values. (Translation: update of shadow registers) 1: Disable UEV. no update event is generated and the shadow registers (ARR, PSC, CCRx) keep their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescalers are reinitialised.
0	CEN	RW	0	Counter enable 0: Disable the counter; 1: Enables the counter. Note: The external clock, gated mode and encoder mode can only work after the CEN bit is set in software. Trigger mode can automatically set the CEN bit in hardware.

13.3.2. TIM1 and TIM8 control registers2 (TIMx_CR2)

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Reserved	CCPC
Reserved	RW	RW	RW	RW	RW	RW	RW	RW	RW			RW	RW	Reserved	RW

Bit	Name	R/W	Reset Value	Function
15	Reserved, always 0			
14	OIS4	RW	0	Output Idle State 4 (OC4 output). See OIS1 bit.
13	OIS3N	RW	0	Output idle state 3 (OC3N output). See the OIS1N bit.
12	OIS3	RW	0	Output idle state 3 (OC3 output). See OIS1 bit.
11	OIS2N	RW	0	Output idle state 2 (OC2N output). See the OIS1N bit.

Bit	Name	R/W	Reset Value	Function
10	OIS2	RW	0	Output idle state 2 (OC2 output). See the OIS1 bit.
9	OIS1N	RW	0	Output Idle state 1 (OC1N output) (Output Idle state 1) 0: When MOE=0, OC1N=0 after deadband; 1: When MOE=0, OC1N=1 after deadband. Note: This bit cannot be modified after LOCK (TIMx_BKR register) level 1, 2 or 3 has been set.
8	OIS1	RW	0	Output Idle state 1 (OC1 output) (Output Idle state 1) 0: When MOE=0, OC1=0 after deadband if OC1N is implemented; 1: When MOE=0, if OC1N is implemented, OC1=1 after deadband. Note: This bit cannot be modified after LOCK (TIMx_BKR register) level 1, 2 or 3 has been set.
7	TI1S	RW	0	TI1 selection 0: The TIMx_CH1 pin is connected to the TI1 input; 1: TIMx_CH1, TIMx_CH2, and TIMx_CH3 pins are connected to the TI1 input after heterodyne.
6:4	MMS	RW	0	Master mode selection These 3 bits are used to select the synchronisation information (TRGO) to be sent to the slave timer in master mode. The possible combinations are as follows: 000: Reset - The UG bit of the TIMx_EGR register is used as the trigger output (TRGO). If the reset is generated by a trigger input (from the mode controller being in reset mode), the signal on the TRGO will have a delay relative to the actual reset. 001: Enable - The counter enable signal CNT_EN is used as the trigger output (TRGO). Sometimes it is necessary to start more than one timer at the same time or to control the enabling of a slave timer over a period of time. The counter enable signal is generated by a logical or of the CEN control bit and the trigger input signal in gated mode. When the counter enable signal is controlled by a trigger input, there is a delay on TRGO unless master/slave mode is selected (see description of the MSM bit in the TIMx_SMCR register). 010: Update - The update event is selected as the trigger input (TRGO). For example, a master timer clock can be used as a prescaler for a slave timer. 011: Compare Pulse - The trigger output sends a positive pulse (TRGO) when the CC1IF flag is to be set (even if it is already high) when a capture or a successful comparison occurs. 100: The Compare - OC1REF signal is used as the trigger output (TRGO). 101: Compare - OC2REF signal is used as trigger output (TRGO). 110: Compare - OC3REF signal is used as trigger output (TRGO). 111: Compare - OC4REF signal is used as trigger output (TRGO). NOTE: The slave timer and ADC clocks must first be enabled to receive signals from the master timer and should not be altered on reception.
3	CCDS	RW	0	Capture/compare DMA selection 0: DMA request for CCx is sent when a CCx event occurs;

Bit	Name	R/W	Reset Value	Function
				1: DMA request for CCx is sent when an update event occurs.
2	CCUS	RW	0	Capture/compare control update selection 0: If the capture/compare control bits are preloaded (CCPC=1), they can only be updated by setting the COM bit; 1: If the capture/compare control bits are preloaded (CCPC=1), they can be updated by setting the COM bit or a rising edge on TRGI. Note: This bit only works for channels with complementary outputs.
1	Reserved, always reads 0.			
0	CCPC	RW	0	Capture/compare preloaded control bits 0: The CCxE, CCxNE and OCxM bits are not preloaded; 1: The CCxE, CCxNE and OCxM bits are preloaded; when this bit is set, they are updated only when the COM bit is set. Note: This bit only works for channels with complementary outputs.

13.3.3. TIM1 and TIM8 Slave Mode Control Registers (TIMx_SMCR)

Address offset:0x08

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
RW	RW	RW		RW				RW	RW			Reserved	RW		

Bit	Name	R/W	Reset Value	Function
15	ETP	RW	0	External trigger polarity This bit selects whether to use ETR or the inverse of ETR as the trigger operation. 0: ETR is not inverted, active high or rising edge; 1: ETR is inverted, active low or falling edge.
14	ECE	RW	0	External clock enable bit This bit enables external clock mode 2 0: disables external clock mode 2; 1: enables external clock mode 2. the counter is driven by any valid edge on the ETRF signal. Note 1: Setting the ECE bit has the same effect as selecting external clock mode 1 and connecting TRGI to ETRF (SMS=111 and TS=111). Note 2: The following slave modes can be used in conjunction with External Clock Mode 2: Reset Mode, Gated Mode, and Trigger Mode; however, the TRGI cannot be connected to the ETRF at this time (the TS bit cannot be '111'). Note 3: When External Clock Mode 1 and External Clock Mode 2 are enabled at the same time, the external clock input is ETRF.
13:12	ETPS	RW	0	External trigger prescaler

Bit	Name	R/W	Reset Value	Function
				The frequency of the external trigger signal ETRP must be at most 1/4 of the TIMxCLK frequency. prescaler can be used to reduce the frequency of ETRP when a faster external clock is input. 00: Turns off the pre-divided frequency; 01: ETRP frequency divided by 2; 10: ETRP frequency divided by 4; 11: ETRP frequency divided by 8.
11:8	ETF	RW	0	External trigger filter These bits define the frequency at which the ETRP signal is sampled and the bandwidth at which the ETRP is digitally filtered. In effect, the digital filter is an event counter that records N events and then generates a jump in the output. 0000: no filter, sampled at fDTS 0001: Sampling frequency fSAMPLING=fCK_INT, N=2 0010: Sampling frequency fSAMPLING=fCK_INT, N=4 0011: Sampling frequency fSAMPLING=fCK_INT, N=8 0100: Sampling frequency fSAMPLING=fDTS/2, N=6 0101: Sampling frequency fSAMPLING=fDTS/2, N=8 0110: Sampling frequency fSAMPLING=fDTS/4, N=6 0111: Sampling frequency fSAMPLING=fDTS/4, N=8 1000: Sampling frequency fSAMPLING=fDTS/8, N=6 1001: Sampling frequency fSAMPLING=fDTS/8, N=8 1010: Sampling frequency fSAMPLING=fDTS/16, N=5 1011: Sampling frequency fSAMPLING=fDTS/16, N=6 1100: Sampling frequency fSAMPLING=fDTS/16, N=8 1101: Sampling frequency fSAMPLING=fDTS/32, N=5 1110: Sampling frequency fSAMPLING=fDTS/32, N=6 1111: Sampling frequency fSAMPLING=fDTS/32, N=8
7	MSM	RW	0	Master/slave mode 0: No effect; 1: The event on the trigger input (TRGI) is delayed to allow perfect synchronisation between the current timer (via TRGO) and its slave timer. This is useful when it is required to synchronise several timers to a single external event.
6:4	TS	RW	0	Trigger selection This 3-bit selection is used to synchronise the trigger input of the counter. 000: Internal trigger 0 (ITR0) 100: Edge detector of TI1 (TI1F_ED) 001: Internal trigger 1 (ITR1) 101: Filtered timer input 1 (TI1FP1) 010: Internal trigger 2 (ITR2) 110: Filtered timer input 2 (TI2FP2) 011: Internal trigger 3 (ITR3) 111: External trigger input (ETRF) For more details on ITRx, see Table 5-1. Note: These bits can only be changed when not used (e.g. SMS=000) to avoid false edge detection when changed.
3	Reserved, always reads 0.			

Bit	Name	R/W	Reset Value	Function
2:0	SMS	RW	0	Slave mode selection When an external signal is selected, the active edge of the trigger signal (TRGI) is related to the polarity of the selected external input (see description of the Input Control Register and Control Register) 000: Off Slave Mode - If CEN=1, the prescaler is driven directly from the internal clock. 001: Encoder Mode 1 - Depending on the level of TI1FP1, the counter counts up/down on the edge of TI2FP2. 010: Encoder Mode 2 - Depending on the level of TI2FP2, the counter counts up/down on the edge of TI1FP1. 011: Encoder Mode 3 - Depending on the input level of another signal, the counter counts up/down on the edges of TI1FP1 and TI2FP2. 100: Reset Mode - The rising edge of the selected trigger input (TRGI) reinitialises the counter and generates a signal to update the registers. 101: Gated Mode - When the trigger input (TRGI) is high, the counter is clocked on. Once the trigger input goes low, the counter is stopped (but not reset). Counter start and stop are controlled. 110: Trigger Mode - The counter is started (but not reset) on the rising edge of the trigger input TRGI, only the start of the counter is controlled. 111: External Clock Mode 1 - The rising edge of the selected trigger input (TRGI) drives the counter. Note: Do not use gated mode if TI1F_EN is selected as the trigger input (TS=100). This is because, TI1F_ED outputs a pulse each time TI1F changes, however the gated mode is to check the level of the trigger input. Note: Do not use uev as the trgo output signal in encoder mode, (i.e. mms cannot be configured to 010)

Table 13-2 TIMx internal trigger connection

Slave timer	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM1	TIM5_TRGO	TIM2_TRGO	TIM3_TRGO	TIM4_TRGO
TIM8	TIM1_TRGO	TIM2_TRGO	TIM4_TRGO	TIM5_TRGO

13.3.4. TIM1 and TIM8 DMA/Interrupt Enable Registers (TIMx_DIER)

Address offset:0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
Reserved	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	Reserved, always 0			
14	TDE	RW	0	Trigger DMA request enable 0: Prohibit triggering of DMA request; 1: triggering of DMA requests is allowed.
13	COMDE	RW	0	COM DMA request enable 0: Disable COM DMA request; 1: allow COM DMA request.
12	CC4DE	RW	0	Capture/Compare 4 DMA request enable 0: Capture/Compare 4 DMA request disabled; 1: Capture/Compare 4 DMA request enable.
11	CC3DE	RW	0	Capture/Compare 3 DMA request enable 0: Capture/Compare 3 DMA request disabled; 1: Capture/Compare 3 DMA request enable.
10	CC2DE	RW	0	Capture/Compare 2 DMA request enable 0: Capture/Compare 2 DMA request disabled; 1: Capture/Compare 2 DMA request enable.
9	CC1DE	RW	0	Capture/Compare 1 DMA request enable 0: Capture/Compare 1 DMA request disabled; 1: Capture/Compare 1 DMA request enable.
8	UDE	RW	0	Update DMA request enable 0: Update DMA request disabled; 1: Update DMA request enabled.
7	BIE	RW	0	Break interrupt enable 0: Disable brake interrupt; 1: brake interrupt enabled.
6	TIE	RW	0	Trigger interrupt enable 0: Trigger interrupt disabled; 1: enable trigger interrupt.
5	COMIE	RW	0	COM interrupt enable 0: COM interrupt is disabled; 1: COM interrupt enabled.
4	CC4IE	RW	0	Capture/Compare 4 interrupt enable 0: Capture/compare 4 interrupts is disabled; 1: Capture/compare 4 interrupt allowed.
3	CC3IE	RW	0	Capture/Compare 3 interrupt enable 0: Capture/Compare 3 interrupt disabled; 1: Capture/Compare 3 interrupt enable.
2	CC2IE	RW	0	Capture/Compare 2 interrupt enable 0: Capture/compare 2 interrupt is disabled; 1: capture/compare 2 interrupt allowed.
1	CC1IE	RW	0	Capture/Compare 1 interrupt enable 0: Capture/compare 1 interrupt is disabled; 1: capture/compare 1 interrupt allowed
0	UIE	RW	0	Update interrupt enable 0: Update interrupt prohibited;

Bit	Name	R/W	Reset Value	Function
				1: update interrupt allowed.

13.3.5. TIM1 and TIM8 Status Registers (TIMx_SR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
Reserved			RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
15:13	Reserved, always 0			
12	CC4OF	RC_W0	0	Capture/Compare 4 overcapture flag Refer to CC1OF for description.
11	CC3OF	RC_W0	0	Capture/Compare 3 overcapture flag Refer to CC1OF for description.
10	CC2OF	RC_W0	0	Capture/Compare 2 overcapture flag Refer to CC1OF for description.
9	CC1OF	RC_W0	0	Capture/Compare 1 overcapture flag This flag can be set by hardware to 1 only if the corresponding channel is configured for input capture. writing 0 clears the bit. 0: No duplicate captures are generated; 1: The state of CC1IF is already '1' when the counter value is captured into the TIMx_CCR1 register.
8	Reserved, always 0.			
7	BIF	RC_W0	0	Break interrupt flag Once the brake input is valid, '1' to this bit by hardware. If the brake input is not valid, the bit can be cleared '0' by software. 0: No brake event is generated; 1: A valid level is detected on the brake input.
6	TIF	RC_W0	0	Trigger interrupt flag It is '1' by hardware to this position when a trigger event occurs (a valid edge is detected on the TRGI input when the slave mode controller is in a mode other than gated mode, or either edge in gated mode). It is cleared '0' by software. 0: No trigger event is generated; 1: Trigger interrupt waiting for response.
5	COMIF	RC_W0	0	COM interrupt flag

Bit	Name	R/W	Reset Value	Function
				This bit is set '1' by hardware as soon as a COM event is generated (when the capture/compare control bits: CCxE, CCxNE, OCxM have been updated). It is cleared '0' by software. 0: No COM event generated; 1: COM interrupt waiting for response.
4	CC4IF	RC_W0	0	Capture/Compare 4 interrupt flag Refer to CC1OF for description
3	CC3IF	RC_W0	0	Capture/Compare 3 interrupt flag Refer to CC1OF for description
2	CC2IF	RC_W0	0	Capture/Compare 2 interrupt flag Refer to CC1OF for description
1	CC1IF	RC_W0	0	Capture/Compare 1 interrupt flag If channel CC1 is configured for output mode: This bit is set by hardware to 1 when the counter value matches the comparison value. except in centre-symmetric mode (refer to the CMS bit in the TIMx_CR1 register). It is cleared '0' by software. 0: No match occurs; 1: The value of TIMx_CNT matches the value of TIMx_CCR1. When the content of TIMx_CCR1 is greater than the content of TIMx_APR, the CC1IF bit goes high under counter overflow in up or up/down counting modes, or counter underflow conditions in down counting mode if channel CC1 is configured for input mode: This bit is set '1' by hardware when a capture event occurs, it is cleared '0' by software or by reading TIMx_CCR1. 0: No input capture is generated; 1: The counter value has been captured (copied) to TIMx_CCR1 (an edge of the same polarity as the one selected is detected on IC1).
0	UIF	RC_W0	0	Update interrupt flag This bit is set '1' by hardware when an update event is generated. It is cleared '0' by software. 0: No update event is generated; 1: update interrupt waiting for response. This bit is set '1' by hardware when the register is updated: - If TIMx_CR1 register UDIS = 0, when the repeat counter value overflows or underflows (update event is generated when repeat counter = 0). - If URS=0 and UDIS=0 of the TIMx_CR1 register, the update event is generated when UG=1 of the TIMx_EGR register is set, and when the counter CNT is reinitialised by software. - If URS=0, UDIS=0 of TIMx_CR1 register, when counter CNT is reinitialised by a trigger event. (Refer to TIM1 and TIM8 Slave Mode Control Register (TIMx_SMCR)).

13.3.6. TIM1 and TIM8 Event Generation Registers (TIMx_EGR)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
Reserved								W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
15:8	Reserved, always 0			
7	BG	W	0	Break generation This bit is set '1' by software to generate a brake event, and is automatically cleared '0' by hardware. 0: No action; 1: Generate a brake event. At this time, MOE=0, BIF=1, if the corresponding interrupt and DMA are turned on, the corresponding interrupt and DMA will be generated.
6	TG	W	0	Trigger generation This bit is set '1' by the software to generate a trigger event, which is automatically cleared '0' by the hardware. 0: No action; 1: TIF=1 in the TIMx_SR register generates the corresponding interrupt and DMA if they are turned on.
5	COMG	W	0	Capture/Compare control update generation This bit is set '1' by software and cleared '0' automatically by hardware. 0: No action; 1: Allows updating of CCxE, CCxNE, OCxM bits when CCPC=1. Note: This bit is only valid for channels with complementary outputs.
4	CC4G	W	0	Generate Capture/Compare 4 generation Refer to the CC1G description.
3	CC3G	W	0	Generate Capture/Compare 3 generation Refer to the CC1G description.
2	CC2G	W	0	Generate Capture/Compare 2 generation Refer to the CC1G description.
1	CC1G	W	0	Capture/Compare 1 generation This bit is set '1' by software to generate a capture/compare event, and is automatically cleared '0' by hardware. 0: No action; 1: Generate a capture/compare event on channel CC1: If channel CC1 is configured as an output: Set CC1IF=1 to generate the corresponding interrupt and DMA if they are turned on. If channel CC1 is configured as an input:

Bit	Name	R/W	Reset Value	Function
				The current counter value is captured to the TIMx_CCR1 register; set CC1IF=1 to generate the corresponding interrupt and DMA if it is switched on. set CC1OF=1 if CC1IF is already 1.
0	UG	W	0	Update generation This bit is set to '1' by software and cleared to '0' automatically by hardware. 0: No action; 1: Re-initialise the counter and generate an update event. Note that the prescaler counter is also cleared '0' (but the prescaler coefficients remain unchanged). The counter is cleared '0' if in centre-symmetric mode or if DIR=0 (counting up); if DIR=1 (counting down) the counter takes the value of TIMx_ARR.

13.3.7. TIM1 and TIM8 Capture/Compare Mode Control Register 1 (TIMx_CCMR1)

Address offset:0x18

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				ICIF[3:0]				ICIPSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

13.3.7.1. Output comparison mode

Bit	Name	R/W	Reset Value	Function
15	OC2CE	RW	0	Output Compare 2 clear enable
14:12	OC2M	RW	0	Output Compare 2 mode
11	OC2PE	RW	0	Output Compare 2 preload enable
10	OC2FE	RW	0	Output Compare 2 fast enable
9:8	CC2S	RW	0	Capture/Compare 2 selection This bit defines the direction of the channel (input/output), and the selection of the input pin: 00: CC2 channel is configured as output; 01: CC2 channel is configured as input, IC2 is mapped on TI2; 10: CC2 channel is configured as input, IC2 is mapped on TI1; 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).

Bit	Name	R/W	Reset Value	Function
7	OC1CE	RW	0	Output Compare 1 clear enable (Output Compare 1 clear enable) 0: OC1REF is unaffected by the ETRF input; 1: Clear OC1REF=0 once the ETRF input is detected high.
6:4	OC1M	RW	0	Output Compare 1 mode These 3 bits define the action of the output reference signal OC1REF, which determines the value of OC1, OC1N. OC1REF is active high, and the active levels of OC1 and OC1N depend on the CC1P and CC1NP bits. 000: Freeze. Comparison between output comparison register TIMx_CCR1 and counter TIMx_CNT does not work for OC1REF; 001: Set channel 1 as the active level when matching. When the value of counter TIMx_CNT is the same as that of capture/comparison register 1 (TIMx_CCR1), force OC1REF to be high. 010: Set channel 1 to invalid level when matching. When the value of counter TIMx_CNT is the same as capture/comparison register 1 (TIMx_CCR1), force OC1REF to be low. 011: Flip-Flop. Flips the level of OC1REF when TIMx_CCR1 = TIMx_CNT. 100: Force to invalid level. Forces OC1REF to be low. 101: Force to valid level. Forces OC1REF to be high. 110: PWM Mode 1- In up count, channel 1 is valid level once TIMx_CNT<TIMx_CCR1, otherwise it is invalid level; in down count, channel 1 is invalid level once TIMx_CNT>TIMx_CCR1 (OC1REF=0), otherwise it is valid level (OC1REF= 1). 111: PWM Mode 2- In up count, channel 1 is invalid level once TIMx_CNT<TIMx_CCR1, otherwise it is valid level; in down count, channel 1 is valid level once TIMx_CNT>TIMx_CCR1, otherwise it is invalid level. Note 1: Once LOCK level is set to 3 (LOCK bit in TIMx_BDTR register) and CC1S=00 (the channel is configured as an output) then this bit cannot be modified. Note 2: In PWM mode 1 or PWM mode 2, the OC1REF level is changed only when the comparison result is changed or when switching from freeze mode to PWM mode in the output comparison mode.
3	OC1PE	RW	0	Output Compare 1 preload enable 0: Disable the preload function of TIMx_CCR1 register, TIMx_CCR1 register can be written at any time, and the newly written value takes effect immediately. 1：Enable the preload function of TIMx_CCR1 register, read/write operation only operates on the preloaded register, and the preloaded value of TIMx_CCR1 is loaded into the current register when the update event arrives. Note 1: This bit cannot be modified once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as an output). Note 2: Only in single pulse mode (OPM=1 in the TIMx_CR1 register), the PWM mode can be used without acknowledging the preload register, otherwise its action is uncertain.
2	OC1FE	RW	0	Output Compare 1 fast enable

Bit	Name	R/W	Reset Value	Function
				This bit is used to speed up the response of the CC outputs to trigger input events. 0: Depending on the value of the counter and CCR1, CC1 operates normally, even if the trigger is open. The minimum delay to activate the CC1 output is 5 clock cycles when the input to the flip-flop has a valid edge. 1: The active edge of the input to the flip-flop acts as if a comparison match has occurred. Therefore, OC is set to the comparison level independent of the comparison result. The delay between the active edge of the sampling flip-flop and the CC1 output is reduced to 3 clock cycles. OCFE only functions when the channel is configured in PWM1 or PWM2 mode.
1:0	CC1S	RW	0	Capture/Compare 1 selection. These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: CC1 channel is configured as input, IC1 is mapped on TI2; 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIMx_CCER register).

13.3.7.2. Input capture mode

Bit	Name	R/W	Reset Value	Function
15:12	IC2F	RW	0	Input capture 2 filter
11:10	IC2PSC	RW	0	Input capture 2 prescaler
9:8	CC2S	RW	0	Capture/Compare 2 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC2 channel is configured as output; 01: CC2 channel is configured as input, IC2 is mapped on TI2; 10: CC2 channel is configured as input, IC2 is mapped on TI1; 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).
7:4	IC1F	RW	0	Input capture 1 filter These bits define the sampling frequency of the TI1 input and the digital filter length. The digital filter consists of an event counter that records N events are recorded to produce a jump in the output:

Bit	Name	R/W	Reset Value	Function
				0000: no filter, sampling at fDTS 1000: sampling frequency fSAMPLING=fDTS/8, N=6 0001: Sampling frequency fSAMPLING=fCK_INT, N=2 1001: Sampling frequency fSAMPLING=fDTS/8, N=8 0010: Sampling frequency fSAMPLING=fCK_INT, N=4 1010: Sampling frequency fSAMPLING=fDTS/16, N=5 0011: Sampling frequency fSAMPLING=fCK_INT, N=8 1011: Sampling frequency fSAMPLING=fDTS/16, N=6 0100: Sampling frequency fSAMPLING=fDTS/2, N=6 1100: Sampling frequency fSAMPLING=fDTS/16, N=8 0101: Sampling frequency fSAMPLING=fDTS/2, N=8 1101: Sampling frequency fSAMPLING=fDTS/32, N=5 0110: Sampling frequency fSAMPLING=fDTS/4, N=6 1110: Sampling frequency fSAMPLING=fDTS/32, N=6 0111: Sampling frequency fSAMPLING=fDTS/4, N=8 1111: Sampling frequency fSAMPLING=fDTS/32, N=8
3:2	IC1PSC	RW	0	Input capture 1 prescaler These 2 bits define the prescaler factor for the CC1 input (IC1). Once CC1E=0 (in the TIMx_CCER register), the prescaler is reset. 00: No prescaler, capture is triggered once for every edge detected on the capture input port; 01: one capture triggered for every 2 events; 10: a capture is triggered every 4 events; 11: a capture is triggered every 8 events.
1:0	CC1S	RW	0	Capture/Compare 1 Selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: The CC1 channel is configured as an output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: CC1 channel is configured as input, IC1 is mapped on TI2; 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIMx_CCER register).

13.3.8. TIM1 and TIM8 Capture/Compare Mode Registers2 (TIMx_CCMR2)

Address offset:0x1C

Reset value:0x0000

Refer to the description of the CCMR1 register above

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

OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

13.3.8.1. Output comparison mode

Bit	Name	R/W	Reset Value	Function
15	OC4CE	RW	0	Output Compare 4 clear enable
14:12	OC4M	RW	0	Output Compare 4 mode
11	OC4PE	RW	0	Output Compare 4 preload enable
10	OC4FE	RW	0	Output Compare 4 fast enable
9:8	CC4S	RW	0	Capture/Compare 4 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC4 channel is configured as output; 01: CC4 channel is configured as input, IC4 is mapped on TI4; 10: CC4 channel is configured as input, IC4 is mapped on TI3; 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC4S is writable only when the channel is off (CC4E=0 in the TIMx_CCER register).
7	OC3CE	RW	0	Output Compare 1clear enable
6:4	OC3M	RW	0	Output Compare 3 mode
3	OC3PE	RW	0	Output Compare 3 preload enable
2	OC3FE	RW	0	Output Compare 3 fast enable
1:0	CC3S	RW	0	Capture/Compare 3 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC3 channel is configured as output; 01: CC3 channel is configured as input, IC3 is mapped on TI3; 10: CC3 channel is configured as input, IC3 is mapped on TI4; 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC3S is writable only when the channel is closed (CC3E=0 in the TIMx_CCER register).

13.3.8.2. Input capture mode

Bit	Name	R/W	Reset Value	Function
15:12	IC4F	RW	0	Input capture 4 filter
11:10	IC4PSC	RW	0	Input capture 4 prescaler
9:8	CC4S	RW	0	Capture/Compare 4 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC4 channel is configured as an output; 01: CC4 channel is configured as input, IC4 is mapped on TI4; 10: CC4 channel is configured as input, IC4 is mapped on TI3; 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC4S is writable only when the channel is off (CC4E=0 in the TIMx_CCER register).
7:4	IC3F	RW	0	Input capture 3 filter
3:2	IC3PSC	RW	0	Input capture 3 prescaler
1:0	CC3S	RW	0	Capture/Compare 3 Selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC3 channel is configured as output; 01: CC3 channel is configured as input, IC3 is mapped on TI3; 10: CC3 channel is configured as input, IC3 is mapped on TI4; 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC3S is writable only when the channel is closed (CC3E=0 in the TIMx_CCER register).

13.3.9. TIM1 and TIM8 Capture/Compare Enable Registers (TIMx_CCER)

Address offset:0x20

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
Reserved		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:14	Reserved, always 0			
13	CC4P	RW	0	Input/Compare 4 output polarity Refer to the description of CC1P.
12	CC4E	RW	0	Capture/Compare 4 output enable Refer to the description of CC1E.
11	CC3NP	RW	0	Input/Capture 3 complementary output polarity Refer to the description of CC1P.
10	CC3NE	RW	0	Capture/Compare 3 output enable Refer to CC1E for description.
9	CC3P	RW	0	Input/Compare 3 output polarity Refer to the description of CC1P.
8	CC3E	RW	0	Capture/Compare 3 output enable Refer to the description of CC1E.
7	CC2NP	RW	0	Input/Capture 2 complementary output polarity Refer to the description of CC1P.
6	CC2NE	RW	0	Capture/Compare 2 output enable Refer to the description of CC1E.
5	CC2P	RW	0	Capture/Compare 2 output polarity Refer to the description of CC1P.
4	CC2E	RW	0	Capture/Compare 2 output enable Refer to the description of CC1E.
3	CC1NP	RW	0	Input/Capture 1 complementary output polarity 0: OC1N active high; 1: OC1N active low. Note: This bit cannot be modified once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 3 or 2 and CC1S=00 (channel configured as output).
2	CC1NE	RW	0	Input/Capture 1 complementary output enable 0: Off - OC1N output is disabled so that the level of OC1N is dependent on the values of the MOE, OSSI, OSSR, OIS1, OIS1N, and CC1E bits. 1: On - The OC1N signal is output to the corresponding output pin, and its output level depends on the values of MOE, OSSI, OSSR, OIS1, OIS1N, and CC1E bit values.
1	CC1P	RW	0	Input/Compare 1 output polarity The CC1 channel is configured as an output: 0: OC1 active high; 1: OC1 active low. The CC1 channel is configured as an input: This bit selects whether IC1 or the inverted signal of IC1 is used as the trigger or capture signal. 0: Not inverted: capture occurs on the rising edge of IC1; when used as an external trigger, IC1 is not inverted.

Bit	Name	R/W	Reset Value	Function
				1: Inverted: Capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted. Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 3 or 2, this bit cannot be modified.
0	CC1E	RW	0	Input/Compare 1 output enable The CC1 channel is configured as an output: 0: Off - OC1 output is disabled, so the OC1 output level is dependent on the values of the MOE, OSSI, OSSR, OIS1, OIS1N, and CC1NE bits. 1: Enable - OC1 signals are output to the corresponding output pins, and its output level depends on the values of MOE, OSSI, OSSR, OIS1, OIS1N, and CC1NE bit values. The CC1 channel is configured as an input: This bit determines whether the counter value can be captured into the TIMx_CCR1 register. 0: Capture disable; 1: capture enable.

Table 13-3 Control bits for complementary output channels OCx and OCxN with brake function

control bits					Output status	
MOEBit	OSSIBit	OSSRBit	CCxEBit	CCxNEBit	OCx Output Status	OCx Output Status
1	x	0	0	0	Output disabled (disconnected from timer) OCx=0, OCx_EN=0	Output disabled (disconnected from timer) OCxN=0, OCxN_EN=0
		0	0	1	Output disabled (disconnected from timer) OCx=0, OCx_EN=0	OCxREF+ polarity. OCxN=OCREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF+ polarity. OCx=OCREF xor CCxP, OCx_EN=1	Output disabled (disconnected from timer) OCxN=0, OCxN_EN=0
		0	1	1	OCxREF+Polarity+Deadband, OCx_EN=1	OCxREF+Polarity+Deadband OCxN_EN=1
		1	0	0	Output disabled (disconnected from timer) OCx=CCxP, OCx_EN=0	Output disabled (disconnected from timer) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Off state (output enabled and invalid level) OCx=CCxP, OCx_EN=1	OCxREF+ polarity. , OCxN=OCREF xor CCxNP, OCxN_EN=1

control bits					Output status	
		1	1	0	OCxREF+ polarity. , OCx=OCREF xor CCxP, OCx_EN=1	Off state (output enabled and invalid level) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCxREF+Polarity+Deadband, OCx_EN=1	OCxREF+Polarity+Deadband, OCxN_EN=1
0	0	x	0	0	Output disabled (disconnected from timer) Asynchronous: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCx_EN=0. If the clock exists: after a dead time OCx=OISx, OCxN=OISxN, assuming that OISx and OISxN do not both correspond to valid levels of OCx and OCxN	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0		
	1		0	1	Off state (output enabled and invalid level) Asynchronous: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCx_EN=1. If the clock is present: after a dead time OCx=OISx, OCxN=OISxN, assuming that OISx and OISxN do not both correspond to valid levels for OCx and OCxN	
	1		1	0		
	1		1	1		

If neither of the 2 outputs of a channel is used (CCxE = CCxNE = 0), then OISx, OISxN, CCxP and CCxNP must be cleared.

Note: The state of the external I/O pins whose pins are connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel states and the GPIO and AFIO registers.

13.3.10. TIM1 and TIM8 counters (TIMx_CNT)

Address offset:0x24

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	Counter value

13.3.11. TIM1 and TIM8 prescalers (TIMx_PSC)

Address offset:0x28

Reset value:0x0000

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

PSC
RW

Bit	Name	R/W	Reset Value	Function
15:0	PSC	RW	0	Prescaler value The counter clock frequency (CK_CNT) is equal to fCK_PSC/ (PSC [15:0] +1). PSC contains the value loaded into the current prescaler register each time an update event is generated; an update event consists of the counter being cleared '0' by the UG bit of TIM_EGR or cleared '0' by a slave controller operating in reset mode.

13.3.12. TIM1 and TIM8 Auto-Reload Registers (TIMx_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	Prescaler value The ARR contains the value to be loaded into the actual auto-reload register. 246/754 Refer to Section 13.3.1: Updates and Actions Regarding the ARR for details. The counter does not operate when the auto-reload value is empty.

13.3.13. TIM1 and TIM8 Repeat Counter Registers (TIMx_RCR)

Address offset:0x30

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:8	Reserved, always reads 0.			
7:0	REP	RW	0	Repetition counter value

Bit	Name	R/W	Reset Value	Function
				With preload enabled, these bits allow the user to set the update rate of the comparison registers (i.e., periodic transfers from the preload registers to the current register); if an update interrupt is allowed to be generated, this also affects the rate at which the update interrupt is generated. Each time the down counter REP_CNT reaches 0, an update event is generated and the counter REP_CNT starts counting again from the REP value. Since REP_CNT only reloads the REP value when the cycle update event U_RC occurs, the new value written to the TIMx_RCR register will only take effect when the next cycle update event occurs. This means that in PWM mode, (REP+1) corresponds to: - In edge-aligned mode, the number of PWM cycles; - In centre-symmetric mode, the number of PWM half-cycles;

13.3.14. TIM1 and TIM8 Capture/Compare Register 1 (TIMx_CCR1)

Address offset:0x34

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR1	RW/ RO	0	Capture/Compare Channel 1 value If the CC1 channel is configured as an output: CCR1 contains the value loaded into the current Capture/Compare 1 register (preloaded value). If the preload function is not selected in the TIMx_CMR1 register (OC1PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare 1 register only when an update event occurs. The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on the OC1 port. If the CC1 channel is configured as an input: CCR1 contains the counter value transferred by the last input capture 1 event (IC1).

13.3.15. TIM1 and TIM8 Capture/Compare Registers2 (TIMx_CCR2)

Address offset:0x38

Reset value:0x0000

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

CCR2
RW/RO

Bit	Name	R/W	Reset Value	Function
15:0	CCR2	RW/ RO	0	Capture/Compare Channel 1 value If the CC2 channel is configured as an output: CCR2 contains the value loaded into the current Capture/Compare 2 register (preloaded value). If the preload function is not selected in the TIMx_CMR2 register (OC2PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare2 register only when an update event occurs. The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on the OC2 port. If the CC2 channel is configured as an input: CCR2 contains the counter value transferred by the last input capture 2 event (IC2).

13.3.16. TIM1 and TIM8 Capture/Compare Registers3 (TIMx_CCR3)

Address offset:0x3C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR3	RW/ RO	0	Capture/Compare 3 value of channel 3 If the CC3 channel is configured as an output: CCR3 contains the value loaded into the current Capture/Compare 3 register (preloaded value). If the preload function is not selected in the TIMx_CMR3 register (OC3PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare3 register only when an update event occurs. The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on the OC3 port. If the CC3 channel is configured as an input: CCR3 contains the counter value transferred by the last input capture 3 event (IC3).

13.3.17. TIM1 and TIM8 Capture/Compare Registers4 (TIMx_CCR4)

Address offset:0x40

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR4	RW/RO	0	Capture/Compare Channel 4 value If the CC4 channel is configured as an output: CCR4 contains the value loaded into the current Capture/Compare 4 register (preloaded value). If the preload function is not selected in the TIMx_CMR4 register (OC4PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare4 register only when an update event occurs. The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on the OC4 port. If the CC4 channel is configured as an input: CCR4 contains the counter value transferred by the last input capture 4 event (IC4).

13.3.18. TIM1 and TIM8 Brake and Deadband Registers (TIMx_BDTR)

Address offset:0x44

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
RW	RW	RW	RW	RW	RW	RW		RW							

Note: Depending on the lock setting, the AOE, BKP, BKE, OSSI, OSSR and DTG [7:0] bits can be write-protected, and it is necessary to configure them the first time the TIMx_BDTR register is written.

Bit	Name	R/W	Reset Value	Function
15	MOE	RW	0	Main output enable This bit is cleared '0' by hardware asynchronously once the brake input is active. Depending on the setting of the AOE bit, this bit can be cleared '0' by software or automatically set to 1. It is only valid for channels configured as outputs. 0: OC and OCN outputs are disabled or forced to an idle state;

Bit	Name	R/W	Reset Value	Function
				1: Enables OC and OCN outputs if the corresponding enable bits (CCxE, CCxNE bits of the TIMx_CCER register) are set. See the TIM1 and TIM8 capture/compare enable registers (TIMx_CCER) for details on OC/OCN enable.
14	AOE	RW	0	Automatic output enable 0: MOE can only be set to '1' by software; 1: MOE can be set to '1' by software or automatically set to '1' at the next update event (if the brake input is disabled). Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to '1', this bit cannot be modified.
13	BKP	RW	0	Brake input polarity (Break polarity) 0: Brake input is active low; 1: Brake input active high. Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to '1', this bit cannot be modified. Note: Any write operation to this bit requires an APB clock delay before it will work.
12	BKE	RW	0	Break enable 0: Disable brake input (BRK and CCS clock failure event); 1: Enable brake input (BRK and CCS clock failure event). Note: When LOCK level 1 is set (LOCK bit in the TIMx_BDTR register), this bit cannot be modified. Note: Any write operation to this bit requires an APB clock delay before it will work.
11	OSSR	RW	0	Off-state selection for Run mode This bit is used when MOE=1 and the channel is a complementary output. The OSSR bit is not present in timers without complementary outputs. Refer to the OC/OCN enable details (TIM1 and TIM8 Capture/Compare Enable Register (TIMx_CCER)). 0: When the timer is not operating, the OC/OCN output is disabled (OC/OCN enable output signal = 0); 1: When the timer is not working, once CCxE=1 or CCxNE=1, first turn on the OC/OCN and output the invalid level, and then set the OC/OCN enable output signal = 1. Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.
10	OSSI	RW	0	Off-state selection for Idle mode This bit is used when MOE=0 and the channel is set to output. Refer to the detailed description of OC/OCN enable (TIM1 and TIM8 Capture/Compare Enable Register (TIMx_CCER)). 0: When the timer is not operating, the OC/OCN output is disabled (OC/OCN enable output signal = 0); 1: When the timer is not working, once CCxE=1 or CCxNE=1, the OC/OCN first outputs its idle level, and then the OC/OCN enable output signal = 1.

Bit	Name	R/W	Reset Value	Function
				Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.
9:8	LOCK	RW	0	Lock configuration This bit provides write protection against software errors. 00: Lock off, registers are not write-protected; 01: Lock level 1, cannot write to the DTG, BKE, BKP, AOE bits of the TIMx_BDTR register and the OISx/OISxN bits of the TIMx_CR2 register; 10: Lockout level 2, cannot write to each bit in lockout level 1, nor can it write to the CC polarity bit (once the channel in question is set to output via the CCxS bit, the CC polarity bit is the CCxP/CCNxP bit of the TIMx_CCER register) and the OSSR/OSSI bits; 11: Lockout level 3, cannot write to each bit in lockout level 2, nor can it write to the CC control bits (once the channel in question is set to output via the CCxS bit, the CC control bits are the OCxM/OCxPE bits of the TIMx_CCMRx register); Note: The LOCK bit can only be written once after a system reset, and once written to the TIMx_BDTR register, its contents are frozen until reset.
7:0	DTG	RW	0	Dead-time generator setup These bits define the duration of the deadband between insertion of complementary outputs. Assume that DT indicates its duration: DTG [7:5] =0xx => DT=DTG [7:0] × Tdtg, Tdtg = TDTS; DTG [7:5] =10x => DT= (64+DTG [5:0]) × Tdtg, Tdtg = 2 × TDTS; DTG [7:5] =110 => DT= (32+DTG [4:0]) × Tdtg, Tdtg = 8 × TDTS; DTG [7:5] =111 => DT= (32+DTG [4:0]) × Tdtg, Tdtg = 16 × TDTS; Example: if TDTS = 125ns (8MHZ), the possible dead time is: 0 to 15875ns, if the step time is 125ns; 16us to 31750ns, if the step time is 250ns; 32us to 63us if the step time is 1us; 64us to 126us, if the step time is 2us; Note: Once the LOCK level (LOCK bits in the TIMx_BDTR register) is set to 1, 2 or 3, these bits cannot be modified.

13.3.19. TIM1 and TIM8 DMA Control Registers (TIMx_DCR)

Address offset:0x48

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:13	Reserved, always 0			
12:8	DBL	RW	0	DMA burst length These bits define the length of the DMA transfer in continuous mode (the timer performs one continuous transfer when a read or write is made to the TIMx_DMAR register), i.e.: defines the number of transfers, which can be half-words (double bytes) or bytes: 00000: 1 transmission 00001: 2 transfers 00010: 3 transfers .... 10001: 18 transmissions Example: Let's consider this transmission: DBL=7, DBA=TIM2_CR1 - If DBL=7, DBA=TIM2_CR1 indicates the address of the data to be transmitted, then the address of the transmission is given by the following equation: (address of TIMx_CR1) + DBA + (DMA index), where DMA index = DBL where (address of TIMx_CR1) + DBA plus 7 gives the address where the data will be written or read, so that the transfer of data will take place in the 7 registers starting from the address (address of TIMx_CR1) + DBA. Depending on the setting of the DMA data length, the following may occur: - If the data is set to half words (16 bits), then the data is transferred to all 7 registers. - If the data is set to byte, the data is still transferred to all 7 registers: the first register contains the first MSB byte, the second register contains the first LSB byte, and so on. Therefore for timers, the user must specify the width of the data to be transferred by the DMA.
7:5	Reserved, always reads 0.			
4:0	DBA	RW	0	DMA base address These bits define the base address of the DMA in continuous mode (when reading or writing to the TIMx_DMAR register), and DBA is defined as the offset from the address where the TIMx_CR1 register is located: 00000: TIMx_CR1. 00001: TIMx_CR2. 00010: TIMx_SMCR. ……..

13.3.20. TIM1 and TIM8 Continuous Mode DMA Addresses (TIMx_DMAR)

Address offset:0x4C

Reset value:0x0000

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

DMAB
RW

Bit	Name	R/W	Reset Value	Function
15:0	DMAB	RW	0	DMA register for burst accesses A read or write to the TIMx_DMAR register results in an access operation to the register where the following address is located: TIMx_CR1 address + DBA + DMA index, where: "TIMx_CR1 address" is the address where control register 1 (TIMx_CR1) is located; "DBA" is the base address defined in the TIMx_DCR register; "DMA Index" is the offset automatically controlled by the DMA, which depends on the DBL defined in the TIMx_DCR register.

Note: When using the DMA continuous transfer function, the value of the CNDTR register of the corresponding channel in the DMA must correspond to the value of DBL in the TIMx_DCR register, otherwise this will not work properly.

13.3.21. TIM1 and TIM8 Register Maps

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
	Read/Write																							rw		rw	rw		rw	rw	rw	rw	rw
	Reset Value	0																						0		0	0		0	0	0	0	0
0x04	TIMx_CR2	Reserved																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS			CCDS	CCUS	Reserved	CCPC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Read/Write																		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
	Reset Value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETPS		ETF				MSM	TS			Reserved	SMS		
	Read/Write																	rw	rw	rw		rw				rw	rw				rw		
	Reset Value	0																0	0	0		0				0	0			0	0		
0x0C	TIMx_DIER	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Read/Write																		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Read/Write																				rc_w0	rc_w0	rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0
	Reset Value	0																			0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	TIMx_EGR	Reserved																								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Read/Write																									w	w	w	w	w	w	w	w
	Reset Value	0																								0	0	0	0	0	0	0	0
0x18	TIMx_CCMR1	Reserved																OC2CE	OC2M		OC2PE	OC2FE	CC2S			OC1CE	OC1M			OC1PE	OC1FE	CC1S	
																		IC2F			IC2PSC					IC1F				IC1PSC			
	Read/Write																	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	
	Reset Value																	0	0	0	0	0	0	0		0	0	0	0	0	0	0	

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset Value	0																0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x1C	TIMx_CCMR2	Reserved																OC4CE	OC4M			OC4PE	OC4FE	CC4S		OC3CE	OC3M			OC3PE	OC3FE	CC3S	
																		IC4F				IC4PSC				IC3F				IC3PSC			
	Read/Write																	rw				rw		rw		rw				rw		rw	
	Reset Value																	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0x20	TIMx_CCER	Reserved																		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Read/Write																			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
	Reset Value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	TIMx_C	Reserved																															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NT																																
	Read/Write																																
	Reset Value																																
0x28	TIMx_PSC																																
	Read/Write																																
	Reset Value																																
0x2C	TIMx_ARR																																
	Read/Write																																
	Reset Value																																

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	TIMx_RCR	Reserved																								REP							
	Read/Write																									rw							
	ResetValue	0																								0							
0x34	TIMx_CCR1	Reserved																CCR1															
	Read/Write																	rw/ro															
	ResetValue	0																0															
0x38	TIMx_CCR2	Reserved																CCR2															
	Read/Write																	rw/ro															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset Value	0																0															
0x3C	TIMx_CCR3	Reserved																CCR3															
	Read/Write																	rw/ro															
	Reset Value	0																0															
0x40	TIMx_CCR4	Reserved																CCR4															
	Read/Write																	rw/ro															
	Reset Value	0																0															
0x44	TIMx_BD	Reserved																MA	BB	OO	LOCK	DTG											

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TR																																
	Read/Write																	rw	rw	rw	rw	rw	rw	rw		rw							
	Reset Value	0																0	0	0	0	0	0	0		0							
0x48	TIMx_DCR	Reserved																			DBL			Reserved		DBA							
	Read/Write																				rw					rw							
	Reset Value	0																			0			0		0							
0x4C	TIMx_DMAR	Reserved																DMAB															
	Read/Write																	rw															
	Reset Va	0																0															

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

14. General-purpose timers (TIM2 to TIM5)

14.1. Introduction

The general purpose timer (TIM2/3/4/5) consists of a 16-bit auto-load counter driven by a programmable prescaler. It is suitable for a variety of uses, including measuring the pulse width of an input signal (input capture), or generating an output waveform (output compare and PWM). Adjustment of pulse width and waveform period from a few microseconds to a few milliseconds is possible using the timer prescaler and the RCC clock control prescaler.

Each general-purpose control timer (TIM2/3/4/5) is completely independent; they do not share any resources. They can be operated synchronously.

14.1.1. Main features of TIM 2/3/4/5

TIM2/3/4/5 timer functions include:

- 16-bit up, down, up/down auto-load counter
- 16-bit programmable (can be modified in real time) prescaler, counter clock frequency with a division factor of any value between 1 and 65536
- Up to 4 independent channels:
  - Input capture
  - Output comparison
  - PWM generation (edge or centre aligned mode)
  - Single pulse mode output
- Synchronisation circuitry to control timer-to-timer interconnections via external signals
- Generate interrupt/DMA when the following events occur:
  - Update: counter overflow up/down, counter initialisation (triggered by software or internal/external)
  - Trigger events (counter start, stop, initialisation or counting triggered internally/externally).
  - Input capture
  - Output comparison
- Supports incremental (quadrature) encoder and Hall sensor circuits for positioning.
- Trigger input as external clock or per-cycle current management

14.1.2. Module block diagram

Figure 100. General-purpose timer block diagram

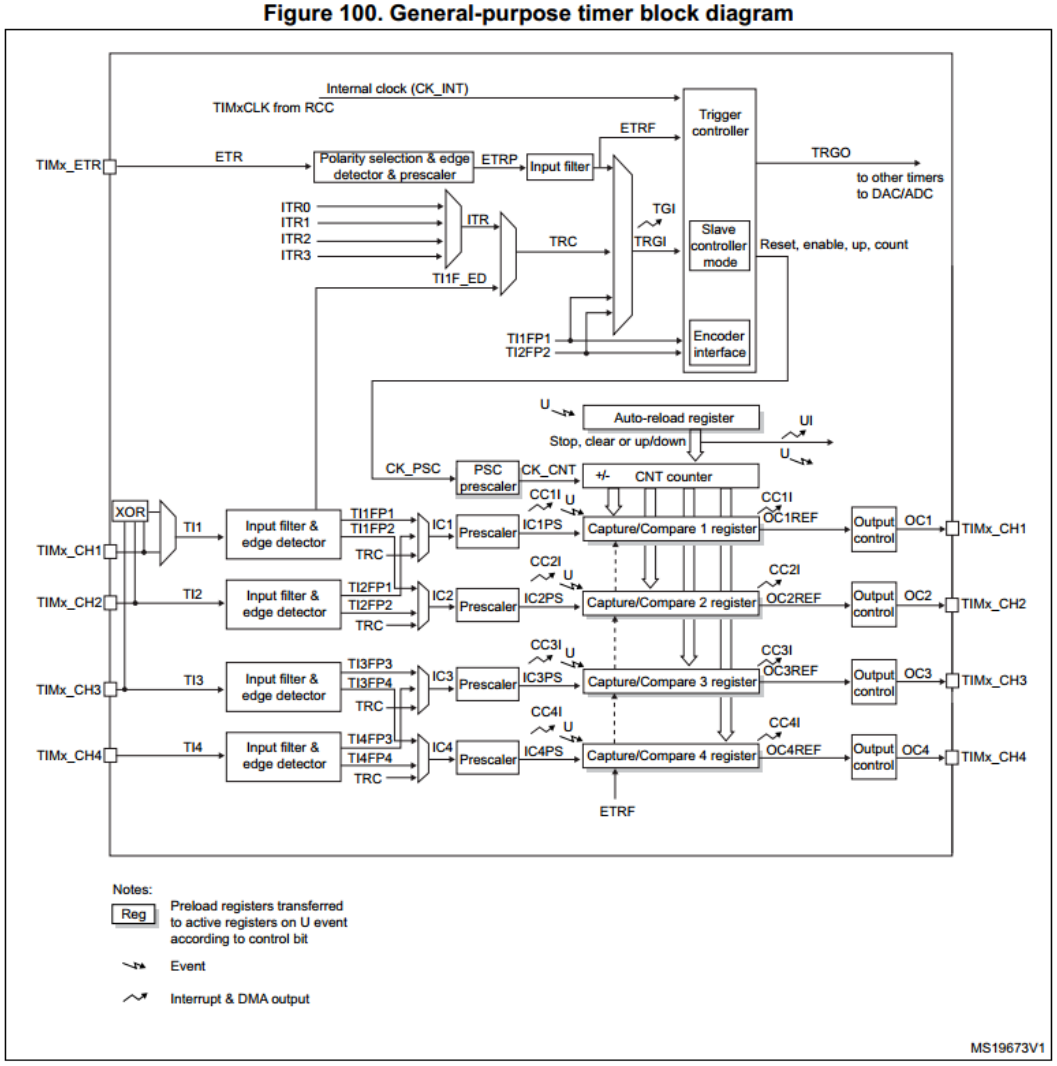


Figure 14-1TIM2/3/4/5 modules

14.2. TIM2/3/4/5 Functional Descriptions

14.2.1. Time base unit

The main part of the programmable general purpose control timer is a 16-bit counter and its associated auto-load register. This counter can count up, count down, or count up and down in both directions. The counter clock can be divided by a prescaler.

The counter, the autoload register and the prescaler register can be read and written by software, and the reads and writes remain valid even when the counter is running.

The time base unit contains:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Autoload Register (TIMx_ARR)

The auto-reload registers are preloaded, and writing or reading an auto-reload register will access its preload registers. Depending on the setting of the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register, the contents of the preload register are transferred to the shadow register either immediately or at each update event (UEV). An update event is generated when the counter reaches

an overflow condition (overflow or underflow) and when the UDIS bit in the TIMx_CR1 register is equal to zero. Update events can also be generated by software and other conditions. The generation of update events for each configuration is described in detail later.

The counter is driven by the prescaler-divided clock output CK_CNT, which is valid for the counter only when the counter enable bit (CEN) in the TIMx_CR1 register is set. (See the description of the Slave Mode Controller for more details on enabling the counter).

Note that the counter starts counting one clock cycle after the CEN bit of the TIMx_CR1 register is set.

14.2.1.1. Prescaler description

The prescaler divides the counter clock frequency by any value between 1 and 65536. It is based on a 16-bit counter controlled by a 16-bit register (in the TIMx_PSC register). Because this control register has a buffer, it can be changed at runtime. The new prescaler parameter will be used when the next update event arrives.

The following figure gives an example of changing the counter parameters while the prescaler is running.

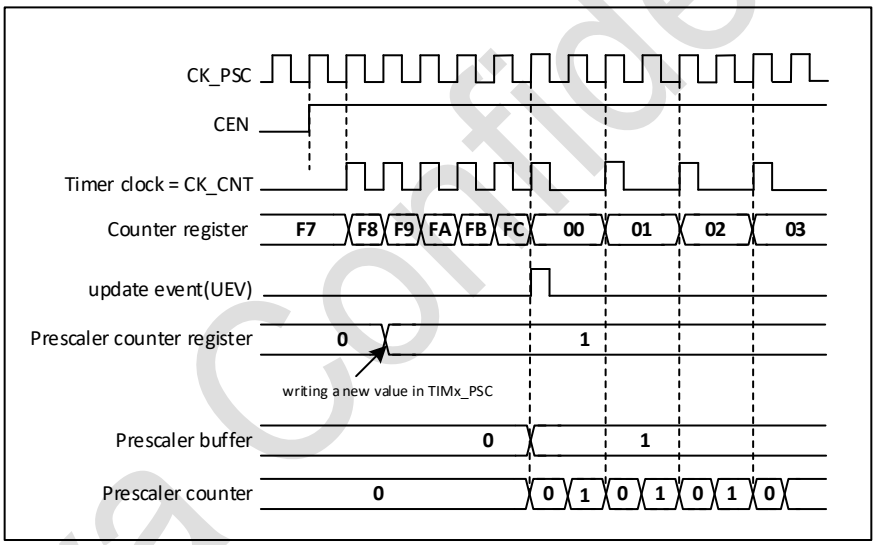


Figure 14-2 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 2

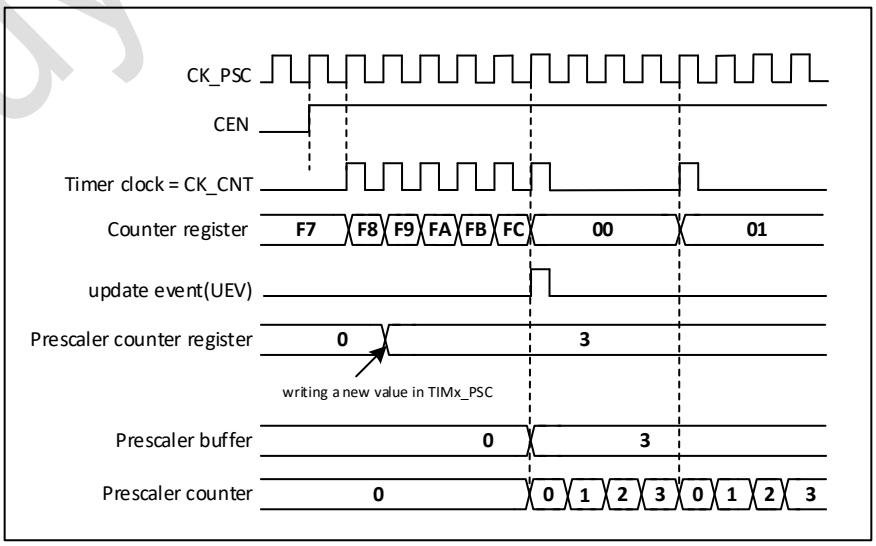


Figure 14-3 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 4

14.2.2. Counter mode

14.2.2.1. Up-counting mode

In count-up mode, the counter counts from 0 to the auto-load value (the contents of TIMx_ARR), then starts counting from 0 again and generates a count-up event.

An update event can be generated each time the counter overflows, and an update event can also be generated by setting the UG bit in the TIMx_EGR register (either in software or using a slave mode controller).

The update event can be disabled by setting the UDIS bit in the TIMx_CR1 register; this prevents the shadow registers from being updated when a new value is written to the preload registers. An update event will not be generated until the UDIS bit is cleared '0'. Even then, the counter will still be cleared '0' when an update event should be generated, and the counter inside the prescaler will also be cleared '0' (but the prescaler value will remain unchanged).

In addition, if the URS bit (Select Update Request Source) in the TIMx_CR1 register is set, an update event UEV can be generated by setting the UG bit, but the UIF flag bit will not be set (i.e. no interrupt or DMA request will be generated). This is to avoid clearing the counter on a capture event and generating both update and capture interrupts.

When an update event occurs, all of the following registers are updated and the hardware simultaneously (based on the URS bit) sets the update flag bit (the UIF bit in the TIMx_SR register):

- The autoload shadow register is reset to the value of the preload register (TIMx_ARR).
- The prescaler buffer is set to the value of the preload register (the contents of the TIMx_PSC register).

The following figure gives some examples of the action of the counter at different clock frequencies when TIMx_ARR = 0x36.

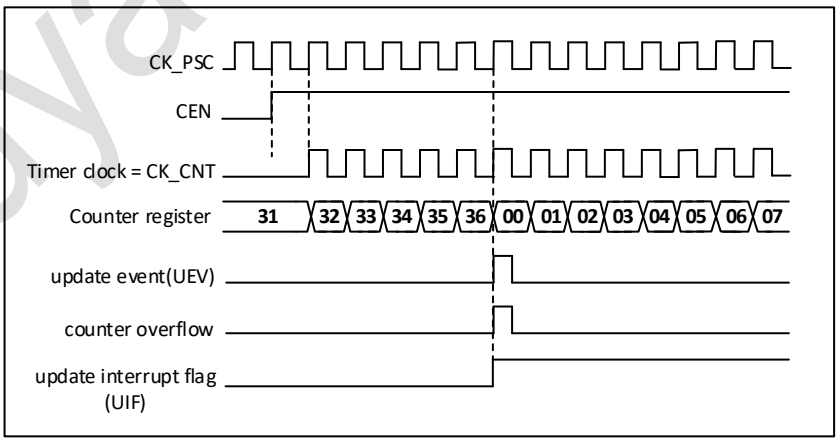


Figure 14-4 Timing diagram of the counter with internal clock division factor of 1

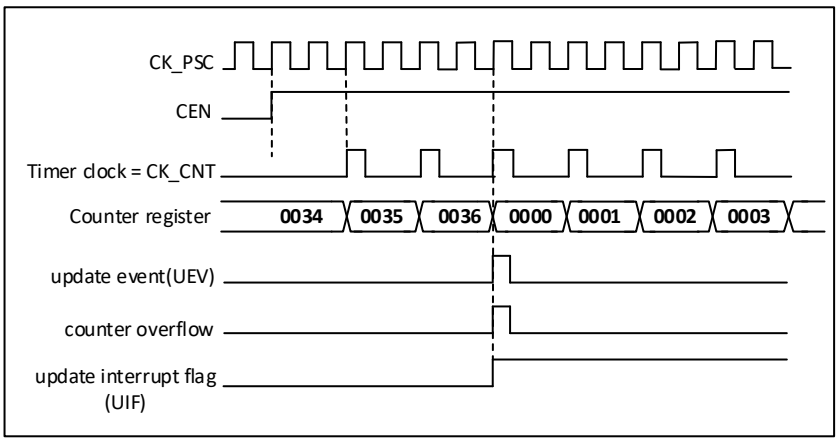


Figure 14-5 Counter Timing Diagram with Internal Clock Division Factor of 2

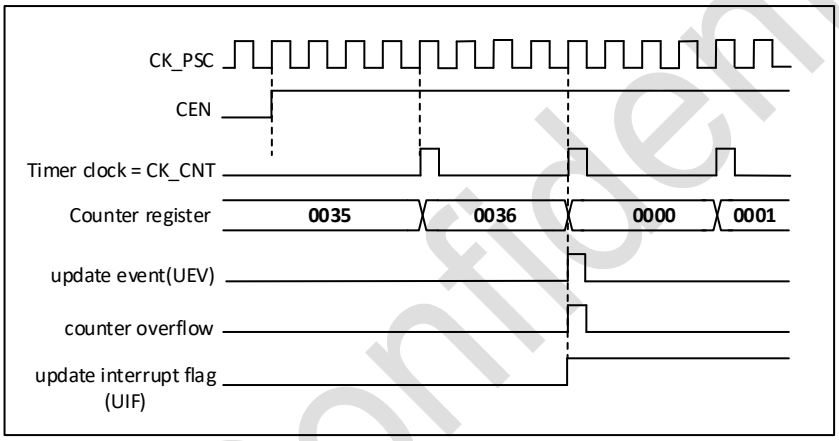


Figure 14-6 Timing diagram of the counter with an internal clock division factor of 4

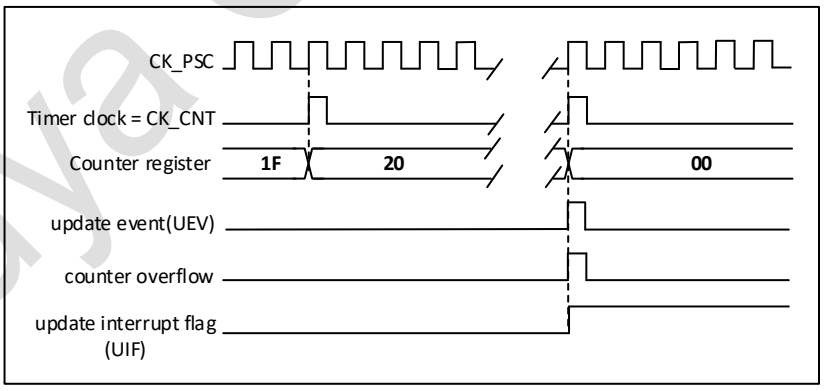


Figure 14-7 Timing diagram of the counter with internal clock division factor N

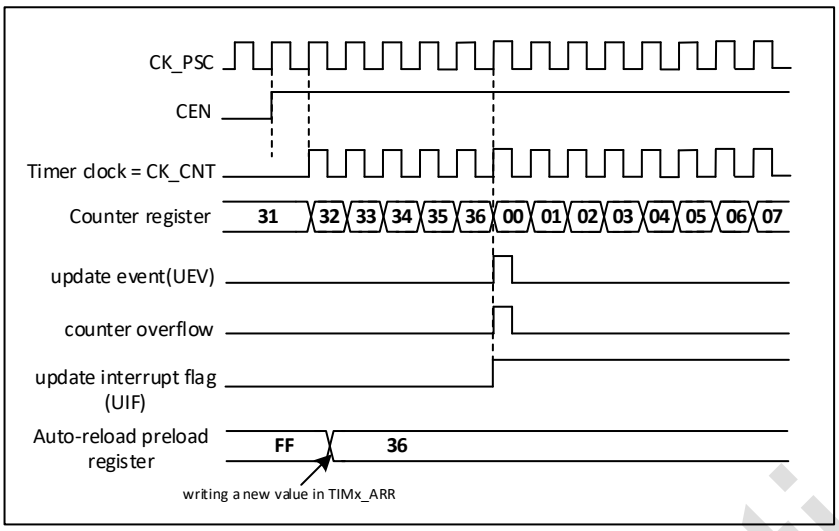


Figure 14-8 Counter timing diagram, update event when ARPE = 0 (no TIMx_ARR preloaded)

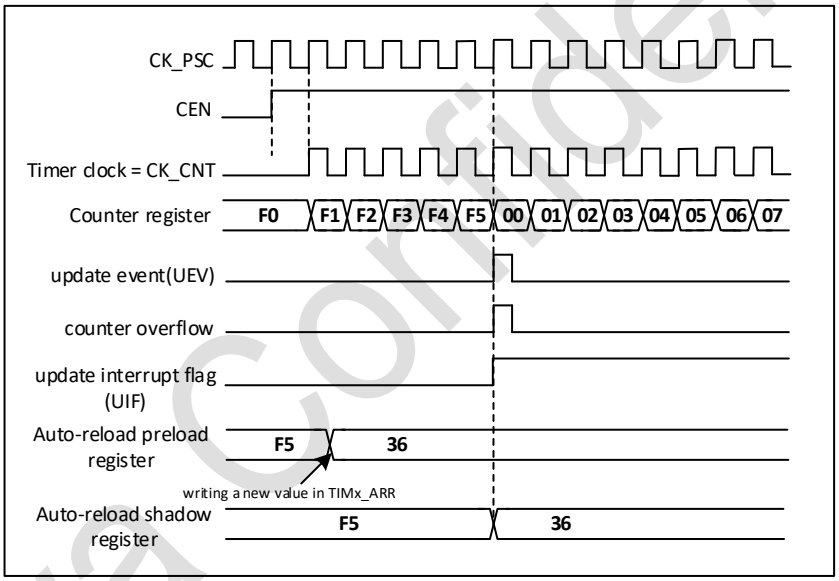


Figure 14-9 Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)

14.2.2.2. Downward counting mode

In count down mode, the counter counts down from the auto-loaded value (the contents of TIMx_ARR) to 0, then restarts from the auto-loaded value and generates a countdown event.

An update event can be generated each time the count overflows, and an update event can also be generated by setting the UG bit in the TIMx_EGR register (either in software or using a slave mode controller).

The update event can be disabled by setting the UDIS bit in the TIMx_CR1 register; this prevents the shadow register from being updated when a new value is written to the preload register. An update event is not generated until the UDIS bit is cleared '0'. Even then, when an update event should be generated, the counter will still restart counting from the current autoload value while the counter inside the prescaler is cleared '0' (but the prescaler coefficient remains unchanged).

In addition, if the URS bit (Select Update Request Source) in the TIMx_CR1 register is set, an update event UEV can be generated by setting the UG bit without setting up the UIF flag bit (i.e., no interrupt or DMA request will be generated), this is to avoid clearing the counters when a capture event occurs, and generating both update and capture interrupts.

When an update event occurs, all following registers are updated and the hardware simultaneously (based on the URS bit) sets the update flag bit (UIF bit in the TIMx_SR register):

- The prescaler buffer is set to the value of the preload register (contents of the TIMx_PSC register).
- The current autoload register is updated with the value of the preload register (contents of the TIMx_ARR register).

Note: The autoload value is updated before the counter is reloaded, so the next cycle will be the expected value.

Here are some examples of counter operation at different clock frequencies when TIMx_ARR = 0x36.

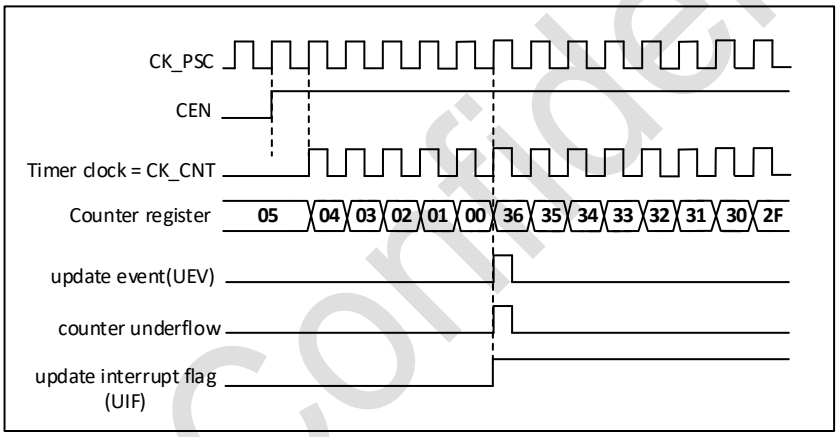


Figure 14-10 Counter Timing Diagram with Internal Clock Division Factor of 1

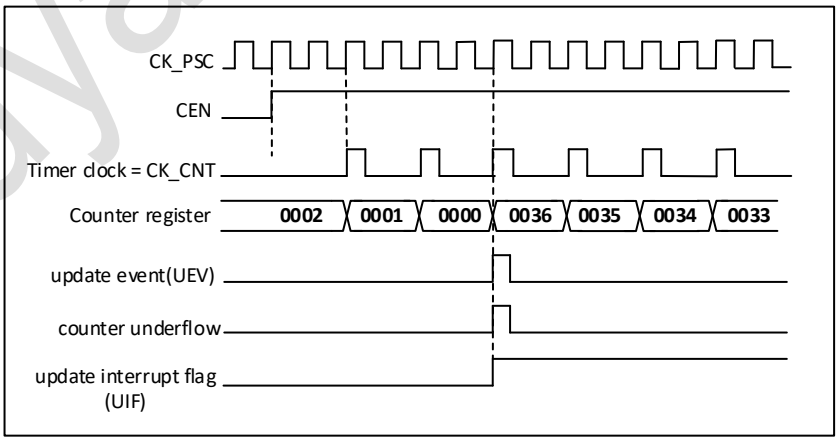


Figure 14-11 Counter Timing Diagram with Internal Clock Division Factor of 2

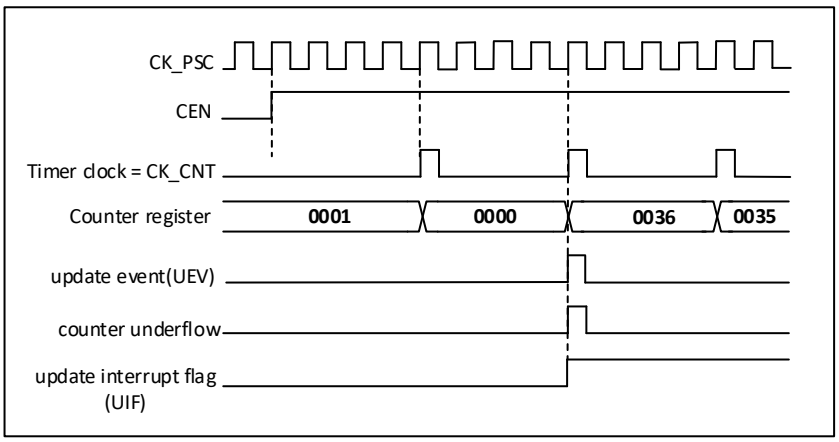


Figure 14-12 Counter Timing Diagram with Internal Clock Division Factor of 4

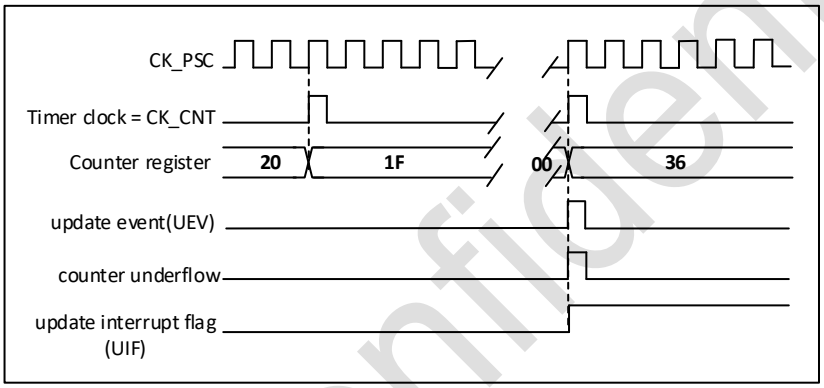


Figure 14-13 Timing diagram of the counter with internal clock division factor N

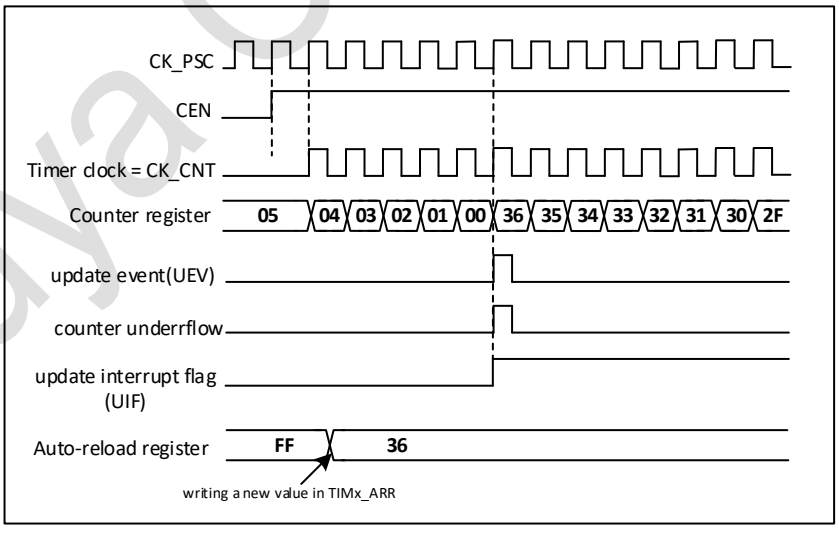


Figure 14-14 Counter Timing Diagram, Update Event When Repeat Counter Not Used

14.2.2.3. Central alignment mode (counting up/down)

In central alignment mode, the counter counts from 0 to the auto-loaded value (TIMx_ARR register) - 1, generates a counter overflow event, then counts down to 1 and generates a counter underflow event; and then counts from 0 again.

Central alignment mode can be obtained by configuring the CMS bit in the TIMx_CR1 register not to be '00'. The Output Compare Flag bit with the channel configured for Output Mode is set during the following types of counting: down counting (Central Alignment Mode 1, CMS='01'), up counting (Central Alignment Mode 2, CMS='10'), when counting up and down (Central Alignment Mode 3, CMS='11').

In this mode, the DIR direction bit in TIMx_CR1 cannot be written. It is updated by hardware and indicates the current count direction.

An update event can be generated on every count overflow and every count underflow; it can also be generated by setting (in software or using a slave mode controller) the UG bit in the TIMx_EGR register. The counter then resumes counting from 0, and the prescaler's internal counter resumes counting from 0 as well.

Setting the UDIS bit in the TIMx_CR1 register disables the update event. This prevents the shadow register from being updated when a new value is written to the preload register. Although the update event is not generated until the UDIS bit is cleared to 0. However, the counter will still continue to count up or down depending on the current auto-reload value.

In addition, if the URS bit (Select Update Request Source) in the TIMx_CR1 register is set, an update event UEV will be generated by setting the UG bit but not setting the UIF flag (and therefore not generating an interrupt and DMA request), this is to avoid clearing the counter when a capture event occurs and generating both an update and capture interrupt.

When an update event occurs, all registers are updated and (depending on the setting of the URS bit) the update flag bit (the UIF bit in the TIMx_SR register) is also set:

- The prescaler buffer is loaded with the value of the preload (TIMx_PSC register).
- The current autoload register is updated to the preloaded value (contents of the TIMx_ARR register).

Note: If an update occurs because of a counter overflow, the auto-reload will be updated before the counter is reloaded, so the next cycle will be the expected value (the counter is loaded with the new value).

Here are some examples of counter operation at different clock frequencies:

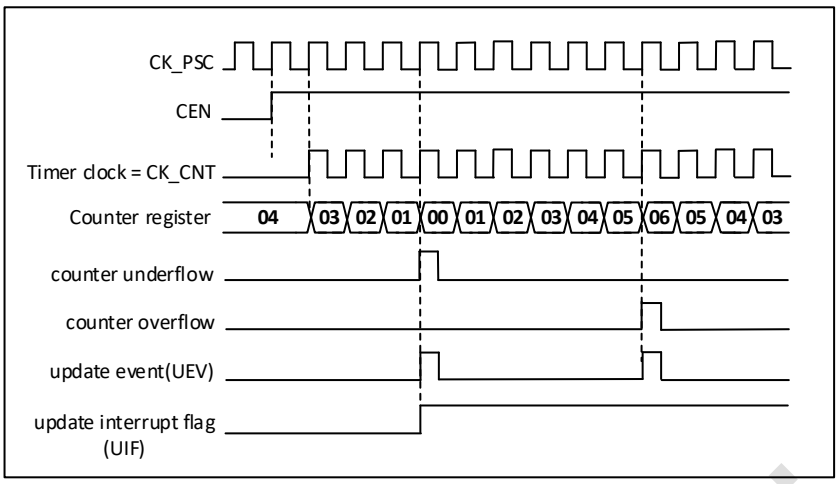


Figure 14-15 Counter Timing Diagram with Internal Clock Division Factor of 1, TIMx_ARR=0x6

Centre alignment mode 1 is used here.

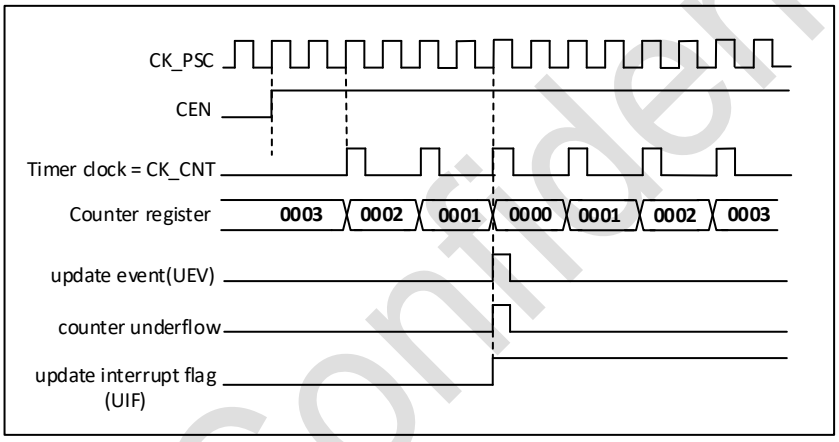


Figure 14-16 Counter Timing Diagram with Internal Clock Division Factor of 2

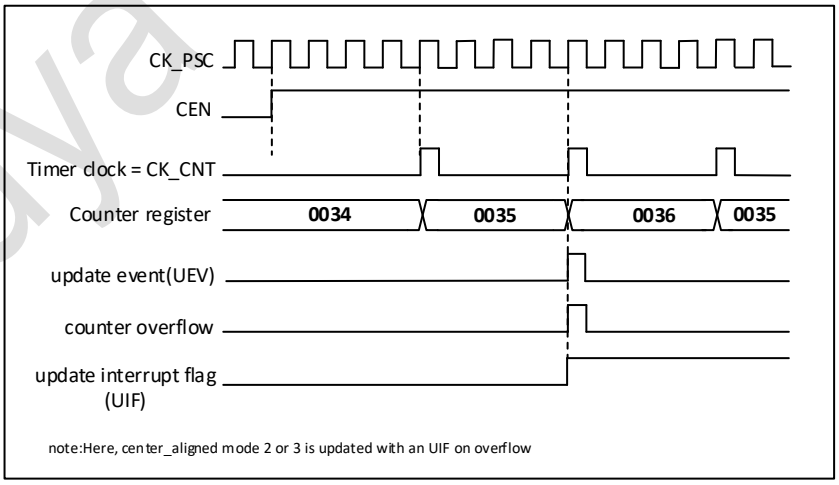


Figure 14-17 Counter Timing Diagram with Internal Clock Division Factor of 4, TIMx_ARR=0x36

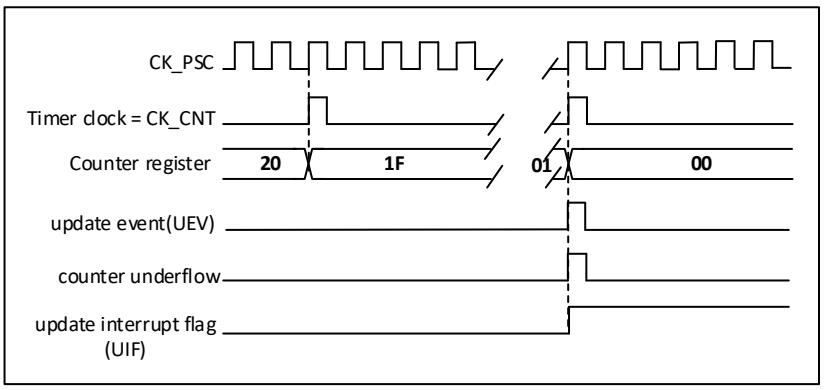


Figure 14-18 Timing diagram of the counter with internal clock division factor N

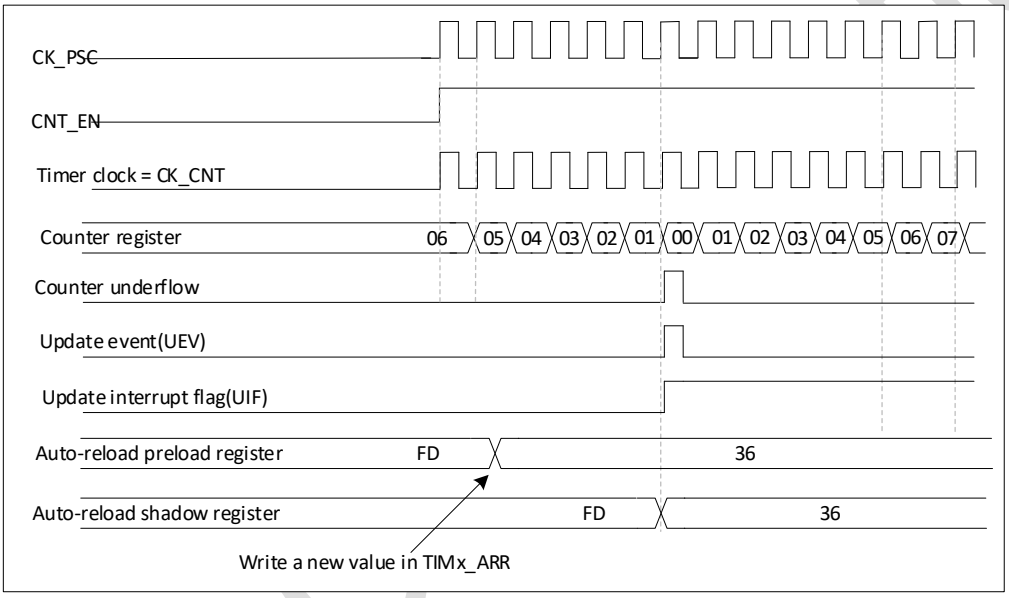


Figure 14-19 Counter Timing Diagram, Update Event at ARPE=1 (Counter Underflow)

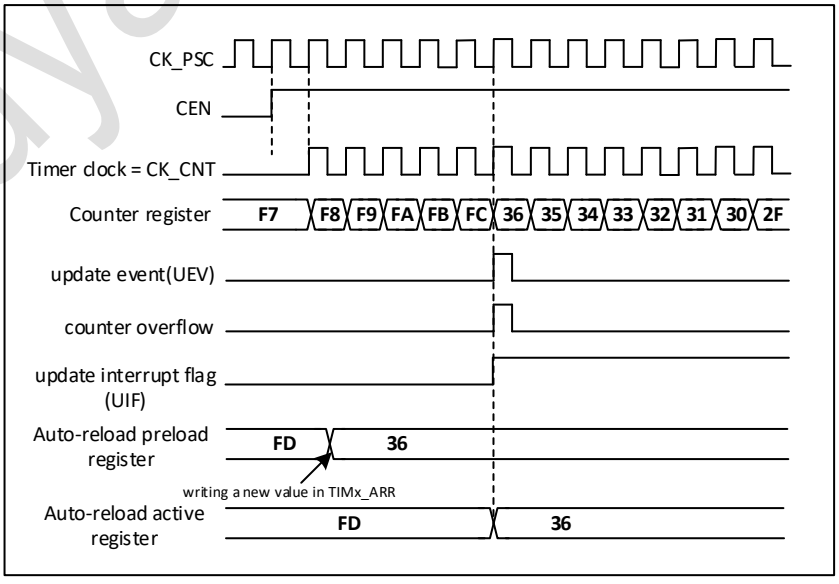


Figure 14-20 Counter Timing Diagram, Update Event at ARPE=1 (Counter Overflow)

14.2.3. Clock selection

The counter clock can be provided by the following clock sources:

Internal clock (CK_INT)

- External clock mode 1: external input pin TIx
- External clock mode 2: external trigger input ETR
- Internal Trigger Input (ITRx): uses one timer as a prescaler for another timer. E.g. it is possible to configure one timer Timer1 and use it as a prescaler for another timer Timer2.

14.2.3.1. Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), the CEN, DIR (TIMx_CR1 register) and UG bits (TIMx_EGR register) are de facto control bits and can only be modified by software (the UG bit is still cleared automatically). As long as the CEN bit is written to '1', the prescaler clock is provided by the internal clock CK_INT.

The following diagram shows the operation of the control circuit and up counter in general mode without prescaler.

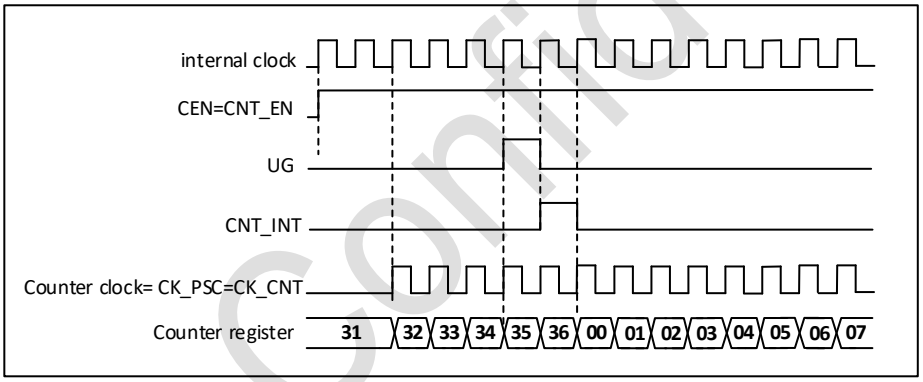


Figure 14-21 Control circuit in general mode with internal clock division factor 1

14.2.3.2. External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count on each rising or falling edge of the selected input.

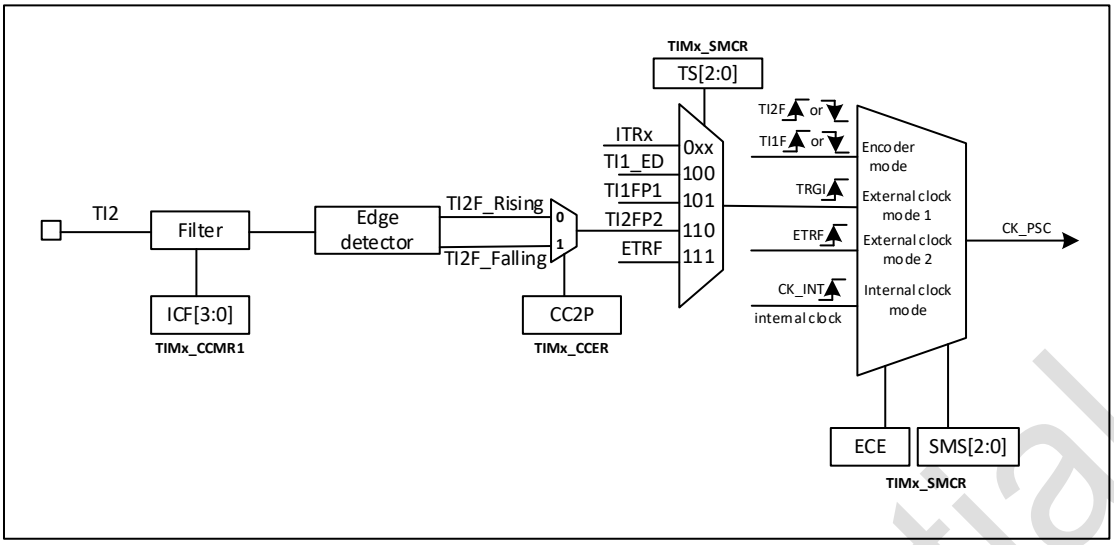


Figure 14-22 TI2 External Clock Connection Example

For example, to configure the counter to count up on the rising edge of the T12 input, use the following steps:

- Configure TIMx_CMR1 register CC2S=01 so that channel 2 detects the rising edge at the TI2 input;
- Configure IC2F [3:0] of the TIMx_CCMR1 register to select the input filter bandwidth (if no filter is required, keep IC2F=0000);
- Configure CC2P=0 of the TIMx_CCER register to select the rising edge polarity;
- Configure SMS=111 of the TIMx_SMCR register to select the timer for external clock mode 1;
- configure TS=110 in the TIMx_SMCR register to select TI2 as the trigger input source;
- Set CEN=1 in the TIMx_CR1 register to start the counter.

Note: The capture prescaler is not used as a trigger, so there is no need to configure it.

When the rising edge occurs at TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge at TI2 and the actual clock of the counter depends on the resynchronisation circuit at the TI2 input.

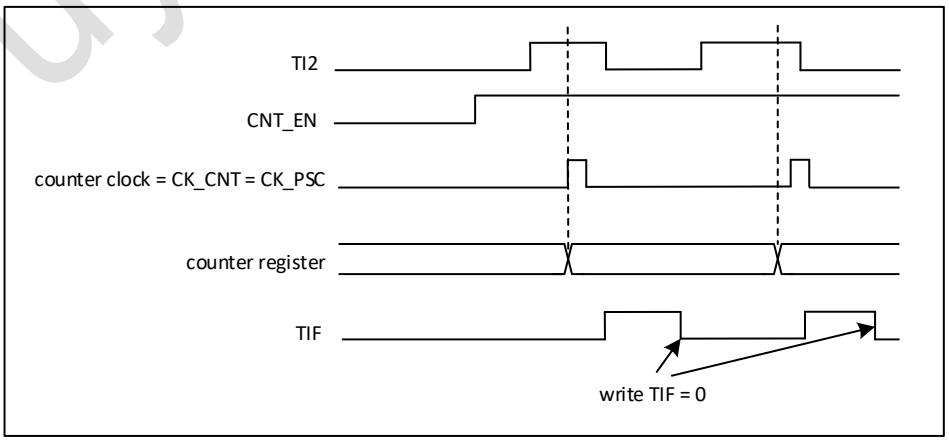


Figure 14-23 Control Circuit in External Clock Mode 1

14.2.3.3. External clock source mode 2

This mode is selected by making ECE=1 in the TIMx_SMCR register, and the counter is able to count on every rising or falling edge of the externally triggered ETR.

The following figure shows the block diagram of the externally triggered input:

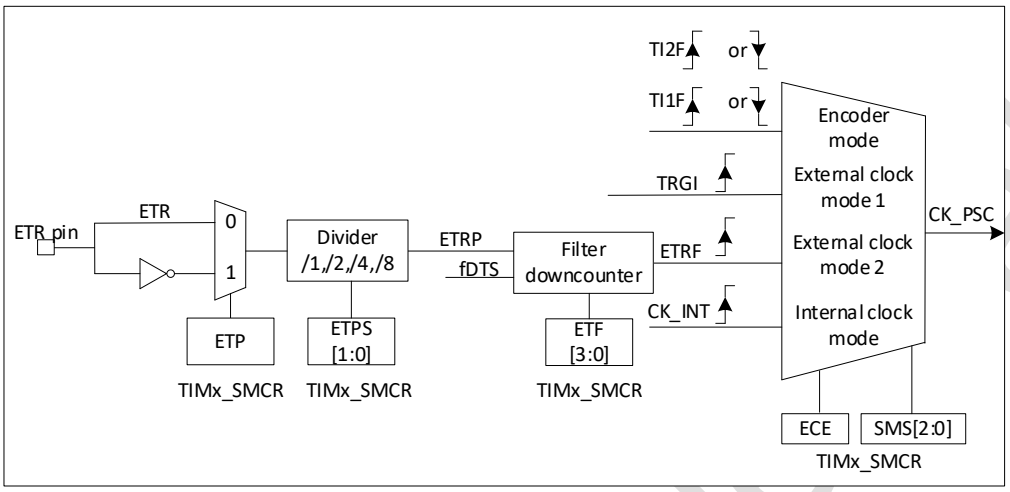


Figure 14-24 External Trigger Input Block Diagram

For example, to configure an up counter that counts every 2 rising edges under ETR, use the following steps:

- No filter is needed in this example, set ETF [3:0] =0000 in the TIMx_SMCR register;
- Set the prescaler, set ETPS [1:0] =01 in the TIMx_SMCR register;
- Select the rising edge of ETR input, set ETP=0 in TIMx_SMCR register;
- Enable external clock mode 2, write ECE=1 in TIMx_SMCR register;
- start the counter, write CEN=1 in the TIMx_CR1 register;
- The counter counts every 2 rising edges of the ETR.

The delay between the rising edge of the ETR and the actual clock of the counter depends on the resynchronisation circuit of the ETRP signal.

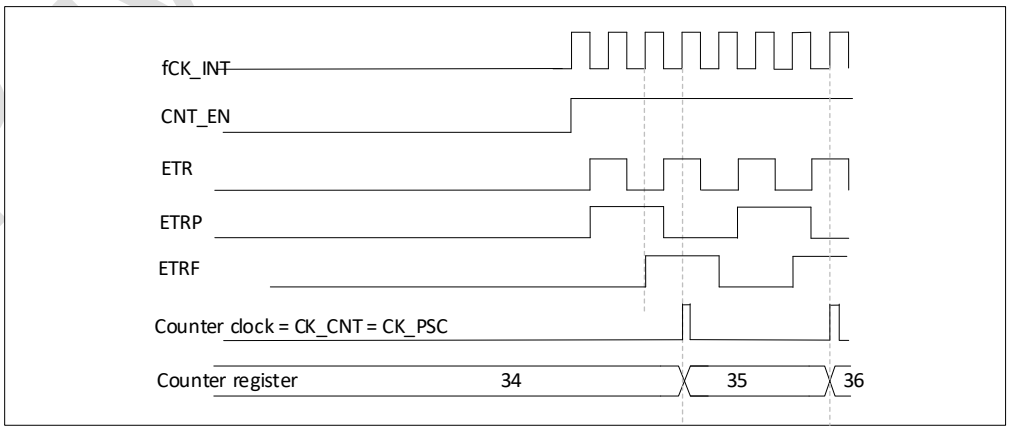


Figure 14-25 Control circuit in external clock mode 2

14.2.4. Capture/Compare Channel

Each capture/compare channel is centred around a capture/compare register (containing shadow registers), including the input portion of the capture (digital filtering, multiplexing and prescaler), and the output portion (comparator and output control).

The input section samples the corresponding TIx input signal and generates a filtered signal, TIxF. An edge monitor with polarity selection then generates a signal (TIxFPx) which can be used as an input trigger from the mode controller or as a capture control. This signal is pre-divided into a capture register (ICxPS). The ICxPS is obtained by dividing the frequency before the capture register.

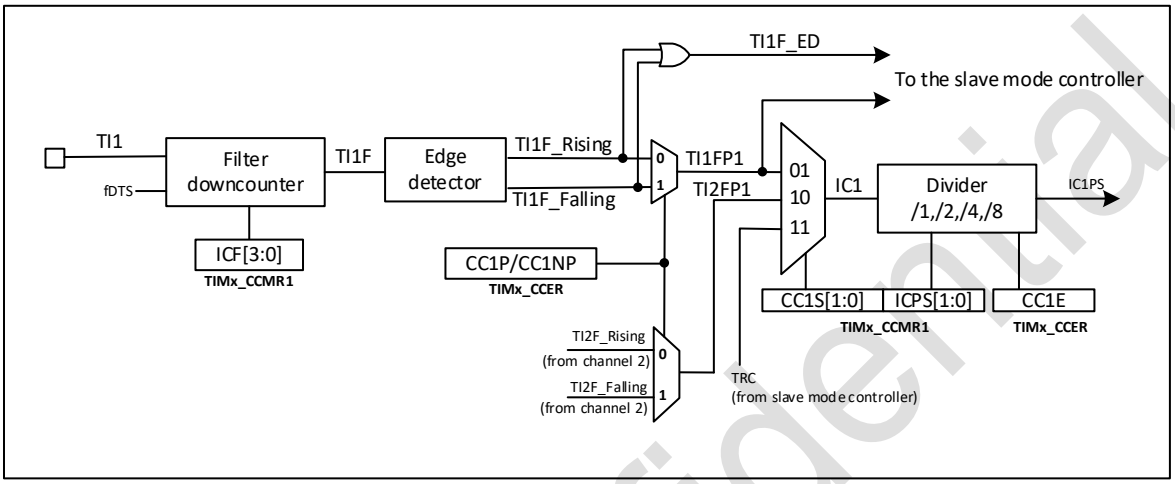


Figure 14-26 Capture/compare channels (e.g. channel 1 input section)

The output section generates an intermediate waveform OCxRef (highly active) as a reference, and the end of the chain determines the polarity of the final output signal.

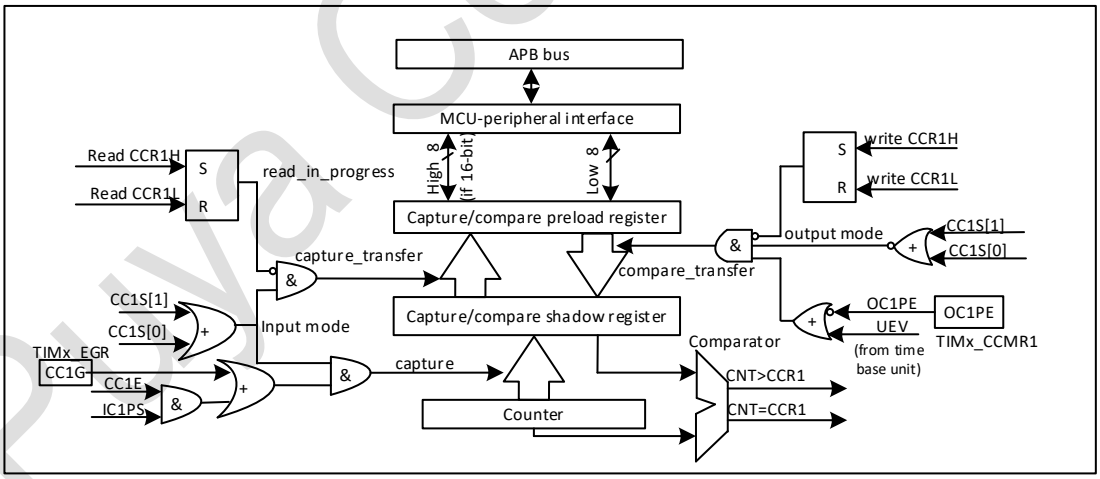


Figure 14-27 Main Circuit for Capture/Compare Channel 1

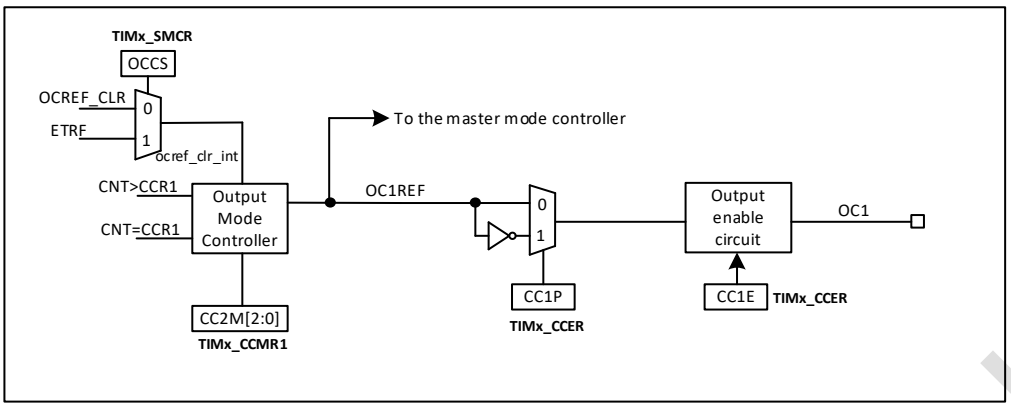


Figure 14-28 Output section of capture/compare channel (channel 1)

The capture/compare module consists of a preloaded register and a shadow register. The read/write process operates only on the preload register.

In capture mode, the capture occurs on the shadow register, which is then copied to the preload register.

In compare mode, the contents of the preload register are copied to the shadow register, and then the contents of the shadow register are compared to the counter.

14.2.5. Input capture mode

In input capture mode, the current value of the counter is latched into the capture/compare register (TIMx_CCRx) when the corresponding edge on the ICx signal is detected. When a capture event occurs, the corresponding CCxIF flag (TIMx_SR register) is set to 1, and an interrupt or DMA request will be generated if an interrupt or DMA operation is open. If the CCxIF flag is already high when a capture event occurs, the overcapture flag CCxOF (TIMx_SR register) is set to 1. CCxIF can be cleared by writing CCxIF=0, or by reading the capture data stored in the TIMx_CCRx register. writing CCxOF=0 clears CCxOF.

The following example shows how to capture the counter value into the TIMx_CCR1 register on the rising edge of the TI1 input as follows:

- Selecting Valid Inputs: TIMx_CCR1 must be connected to the TI1 input, so write CC1S=01 to the TIMx_CCR1 register, as long as CC1S is not '00', the channel is configured as an input and the TIMx_CCR1 register becomes read-only.
- Configure the input filter for the desired bandwidth based on the characteristics of the input signal (i.e., when the input is TIx, the input filter control bit is the ICxF bit in the TIMx_CMRx register). Assuming that the input signal dithers over a period of up to 5 internal clock cycles, we have to configure the filter with a bandwidth longer than 5 clock cycles; we can therefore (at the fDTS frequency) sample the input 8 times in a row in order to confirm a true edge shift on TI1, i.e., by writing IC1F=0011 in the TIMx_CCMR1 register.
- Select the valid transition edge on the TI1 channel by writing CC1P=0 (set to rising edge) in the TIMx_CCER register.
- Configure the input prescaler. In this example, we want the capture to occur at every valid level transition moment, so the prescaler is disabled (write IC1PS=00 in the TIMx_CMR1 register).

- Set CC1E=1 of the TIMx_CCER register to allow the capture counter value to be captured into the capture register.
- If desired, allow related interrupt requests by setting the CC1IE bit in the TIMx_DIER register, and also allow DMA requests by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- When a valid level transition is generated, the counter value is transferred to the TIMx_CCR1 register.
- The CC1IF flag bit is set (interrupt flag). When at least 2 consecutive captures occur and CC1IF has not been cleared, CC1OF is also set to 1.
- If the CC1IE bit is set, an interrupt is generated.
- If the CC1DE bit is set, a DMA request is also generated.
- In order to handle overcatch, it is recommended to read the data before the overcatch flag is set, this is to avoid losing capture overflow information that may be generated after the capture overflow flag is read and before the data is read.

Note: Input capture interrupts and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

14.2.6. PWM Input Mode

This mode is a special case of the Input Capture mode and operates the same as the Input Capture mode except for the following differences:

- The two ICx signals are mapped to the same TIx input.
- The 2 ICx signals are edge-valid, but of opposite polarity.
- One of the TIxFP signals is used as the trigger input signal and the slave mode controller is configured in reset mode.

For example, the user can measure the period (TIMx_CCR1 register) and duty cycle (TIMx_CCR2 register) of the PWM signal input to TI1 as follows (depending on the frequency of CK_INT and the value of the prescaler).

- Select the valid input of TIMx_CCR1: set CC1S=01 of TIMx_CMR1 register (TI1 selected).
- Select valid polarity of TI1FP1 (used to capture data into TIMx_CCR1 and clear the counter): set CC1P=0 (valid on rising edge).
- Select valid input for TIMx_CCR2: set CC2S=10 of TIMx_CMR1 register (TI1 selected).
- Select valid polarity of TI1FP2 (capture data to TIMx_CCR2): set CC2P=1 (falling edge valid).
- Select valid trigger input signal: set TS=101 in TIMx_SMCR register (select TI1FP1).
- Configure slave mode controller to reset mode: set SMS=100 in TIMx_SMCR.
- Enable capture: set CC1E=1 and CC2E=1 in the TIMx_CCER register.

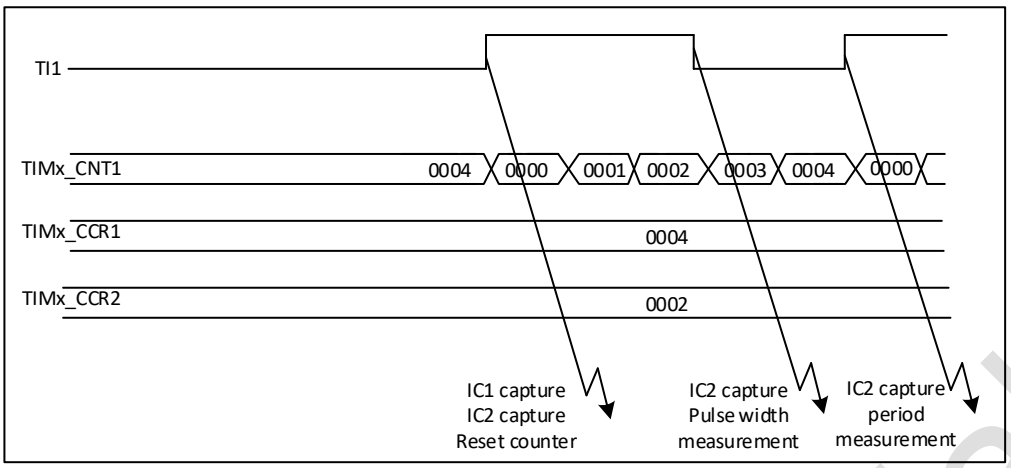


Figure 14-29 PWM Input Mode Timing

Since only TI1FP1 and TI2FP2 are connected to the Slave Mode Controller, only the TIMx_CH1/TIMx_CH2 signals can be used for PWM input mode.

14.2.7. Forced output mode

In output mode (CCxS=00 in TIMx_CCMRx register), the output compare signals (OCxREF and corresponding OCx) can be forced to valid or invalid state by software directly, independent of the comparison result between the output compare register and counter.

The output compare signal (OCxREF/OCx) can be forced to a valid state by setting the corresponding OCxM=101 in the TIMx_CMRx register. In this way, OCxREF is forced high (OCxREF is always active high), while OCx gets the signal of CCxP with opposite polarity.

For example: CCxP=0 (OCx active high), then OCx is forced high.

Setting OCxM=100 in the TIMx_CMRx register forces the OCxREF signal low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter is still in progress and the corresponding flags are modified. Therefore, corresponding interrupts and DMA requests are still generated. This will be described in the Output Compare Mode section below.

14.2.8. Output comparison mode

This function is used to control an output waveform or to indicate that a given period of time has elapsed.

When the counter and the capture/compare registers have the same contents, the output compare function does the following:

- Outputs the values defined by the output compare mode (OCxM bit in the TIMx_CCMRx register) and output polarity (CCxP bit in the TIMx_CCER register) to the corresponding pins. The output pin can hold its level (OCxM=000), be set to an active level (OCxM=001), be set to an inactive level (OCxM=010), or be flipped (OCxM=011) when comparing matches.
- Set the flag bit in the interrupt status register (CCxIF bit in the TIMx_SR register).
- An interrupt is generated if the corresponding interrupt mask (CCxIE bit in the TIMx_DIER register) is set.
- A DMA request is generated if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register selects the DMA request function).

The need for the TIMx_CCRx registers to use preloaded registers can be selected by configuring the OCxPE bit in TIMx_CMRx.

In output compare mode, the update event UEV has no effect on the OCxREF and OCx outputs. The synchronisation can be done with an accuracy of up to one count cycle of the counter. Output compare mode (in single pulse mode) can also be used to output a single pulse.

Configuration steps for Output Compare Mode:

- Selects the counter clock (internal, external, prescaler).
- Write the corresponding data to the TIMx_ARR and TIMx_CCRx registers.
- Set the CCxIE bit if an interrupt request is to be generated.
- Select the output mode, e.g.:
  - Flip the output pin of OCx when the counter is required to match CCRx, set OCxM = 011
  - Set OCxPE = 0 to disable the preload register.
  - Set CCxP = 0 to select polarity active high.
  - Set CCxE = 1 to enable outputs.
- Set the CEN bit of the TIMx_CR1 register to start the counter

The TIMx_CCRx registers can be updated by software at any time to control the output waveform, provided that the preloaded registers are not used (OCxPE='0', otherwise the shadow registers of TIMx_CCRx can only be updated when the next update event occurs). An example is given in the figure below.

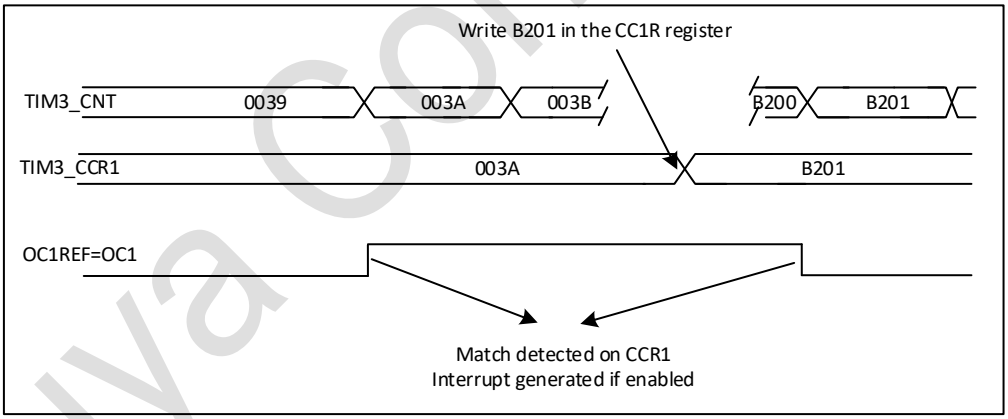


Figure 14-30 Output Compare Mode, Flip OC1

14.2.9. PWM mode

Pulse width modulation mode can generate a signal with a frequency determined by the TIMx_ARR register and a duty cycle determined by the TIMx_CCRx register.

Writing '110' (PWM mode 1) or '111' (PWM mode 2) to the OCxM bit in the TIMx_CMRx register enables each OCx output channel to be set independently to generate a PWM all the way through. this must be done by setting the The corresponding preload registers must be enabled by setting the OCxPE bit of the TIMx_CMRx register, and finally the ARPE bit of the TIMx_CR1 register must be set to enable the auto-reloaded preload registers (in count-up or centre-symmetric mode).

The preloaded registers are transferred to the shadow registers only when an update event occurs, so the user must initialise all registers by setting the UG bit in the TIMx_EGR register before the counter starts counting.

The polarity of the OCx can be set by software in the CCxP bit in the TIMx_CCER register, which can be set to active high or active low. the CCxE bit in the TIMx_CCER register controls the OCx output enable. See the description of the TIMx_CCERx register for details.

In PWM mode (Mode 1 or Mode 2), TIMx_CNT and TIMx_CCRx are always being compared, (based on the counter count direction) to determine compliance with TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx.

However, in order to be consistent with the function of OCREF_CLR (an external event on the ETR signal can clear OCxREF before the next PWM cycle), the OCxREF signal can only be generated under the following conditions:

- When the result of the comparison changes, or
- When the output comparison mode (OCxM bit in the TIMx_CMRx register) switches from "frozen" (no comparison, OCxM='000') to some PWM mode (OCxM='110' or '111'). '110' or '111').

This allows the PWM output to be forced by software during operation.

Depending on the state of the CMS bit in the TIMx_CR1 register, the timer is able to generate either an edge-aligned PWM signal or a centre-aligned PWM signal.

14.2.9.1. PWM Edge Alignment Mode

- Upward Count Configuration

Upward counting is performed when the DIR bit in the TIMx_CR1 register is low.

The following is an example for PWM mode 1. The PWM reference signal OCxREF is high when TIMx_CNT<TIMx_CCRx and low otherwise. If the comparison value in TIMx_CCRx is greater than the auto-reload value (TIMx_ARR), OCxREF remains '1'. If the comparison value is 0, OCxREF remains '0'. The following figure shows an example of an edge-aligned PWM waveform with TIMx_ARR=8.

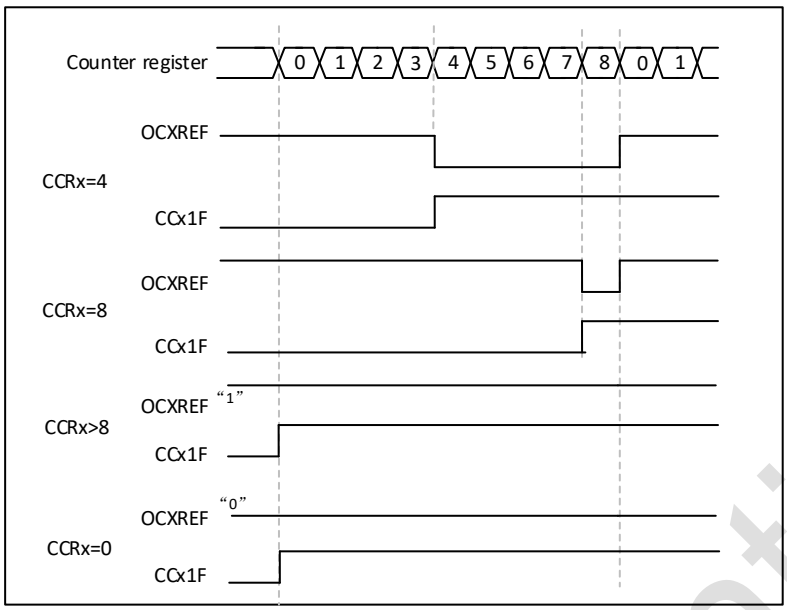


Figure 14-31 Edge-aligned PWM waveform (ARR=8)

- Configuration for Downward Counting

Downward counting is performed when the DIR bit of the TIMx_CR1 register is high.

In PWM mode 1, the reference signal OCxREF is low when TIMx_CNT > TIMx_CCRx and high otherwise. If the comparison value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, OCxREF remains '1'. A 0% PWM waveform cannot be generated in this mode.

14.2.9.2. PWM Central Alignment Mode

Central alignment mode when the CMS bit in the TIMx_CR1 register is not '00' (all other configurations of the CMS bit have the same effect on the OCxREF/OCx signals). Depending on the CMS bit setting, the compare flag can be set to 1 when the counter is counting up, 1 when the counter is counting down, or 1 when the counter is counting both up and down. The Count Direction Bit (DIR) in the TIMx_CR1 register is updated by hardware; do not modify it in software.

The following figure gives some examples of centre-aligned PWM waveforms

- TIMx_ARR=8
- PWM mode 1
- CMS=01 in TIMx_CR1 register sets the compare flag when the counter counts down in Central Alignment Mode 1.

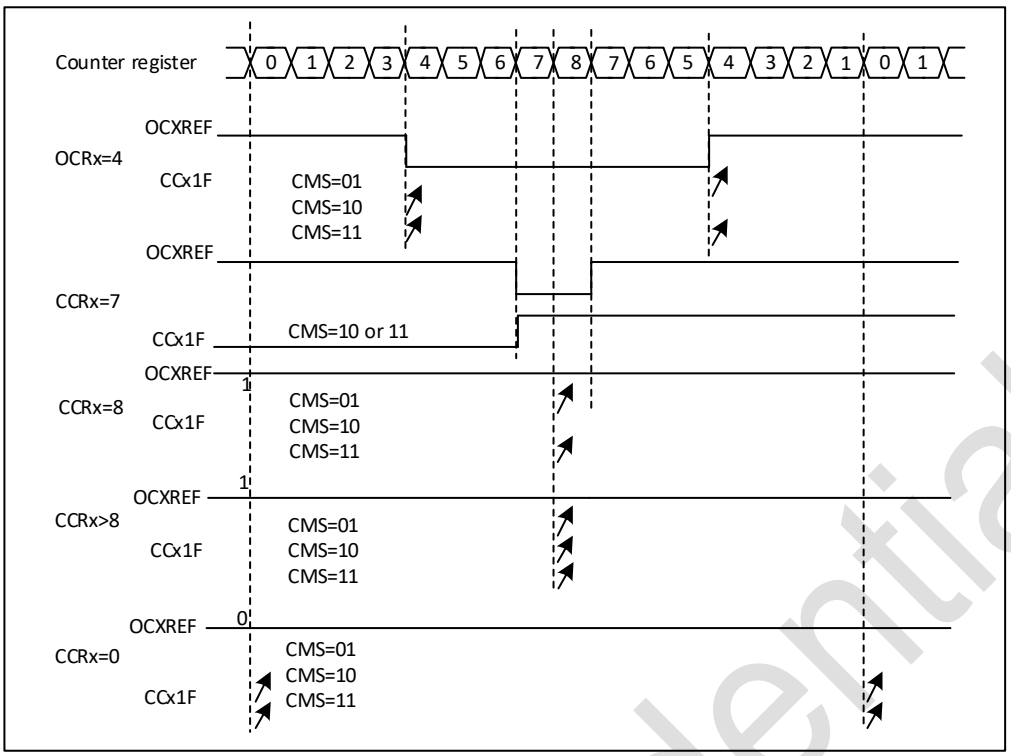


Figure 14-32 Centrally aligned PWM waveform (APR=8)

Hints for using Central Alignment Mode:

- The current count up/down configuration is used when entering central alignment mode; this means that whether the counter counts up or down depends on the current value of the DIR bit in the TIMx_CR1 register. Also, do not modify both the DIR and CMS bits via software.
- It is not recommended to rewrite the counter when running in centre-aligned mode, as this can produce unpredictable results. In particular:
  - If the value written to the counter is greater than the auto-reload value (TIMx_CNT>TIMx_ARR), the direction will not be updated.

For example, if the counter is counting up, it continues to count up. If a value of 0 or TIMx_ARR is written to the counter, the direction is updated, but no update event UEV is generated.

- The safest way to use the central alignment mode is to generate a software update (set the UG bit in the TIMx_EGR bit) before starting the counter and not to modify the counter value while the count is in progress.

14.2.10. Clear the OCxREF signal on an external event

For a given channel, setting the corresponding OCxCE bit in the TIMx_CMRx register to '1' is able to pull the OCxREF signal low with a high level on the ETRF input, and the OCxREF signal will remain low until the next update event, UEV, occurs.

This function can only be used in output compare and PWM modes, not in force-set mode.

For example, the OCxREF signal can be coupled to the output of a comparator for current control. In this case, the ETR must be configured as follows:

- The externally triggered prescaler must be off: ETPS [1:0] =00 in the TIMx_SMCR register.
- External clock mode 2 must be disabled: ECE=0 in the TIMx_SMCR register.

- The External Trigger Polarity (ETP) and External Trigger Filter (ETF) can be configured as required.

The following figure shows the action of the OCxREF signal when the ETRF input goes high, corresponding to different OCxCE values. In this example, the timer TIMx is placed in PWM mode.

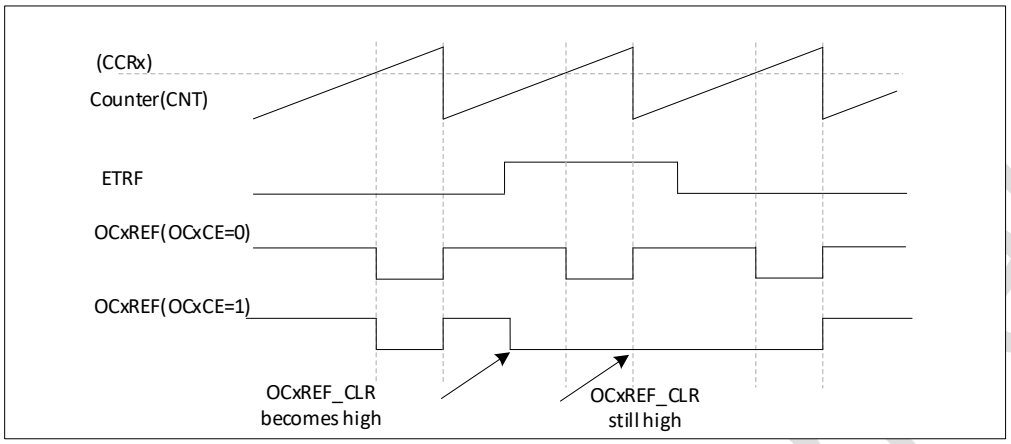


Figure 14-33Clear OCxREF for TIMx

14.2.11. Single pulse mode

One-pulse mode (OPM) is a special case of one of the many modes described previously. This mode allows the counter to respond to an excitation and generate a pulse with a programmable pulse width after a programmable delay.

The counter can be activated from the mode controller to generate a waveform in either output comparison mode or PWM mode. Setting the OPM bit in the TIMx_CR1 register will select single pulse mode, which allows the counter to automatically stop when the next update event, UEV, is generated.

A pulse is generated only when the comparison value is different from the initial value of the counter. Before starting (when the timer is waiting to be triggered), it must be configured as follows:

- Upward counting mode: counter CNT < CCRx ≤ ARR (in particular, 0 < CCRx).
- downward counting mode: counter CNT > CCRx.

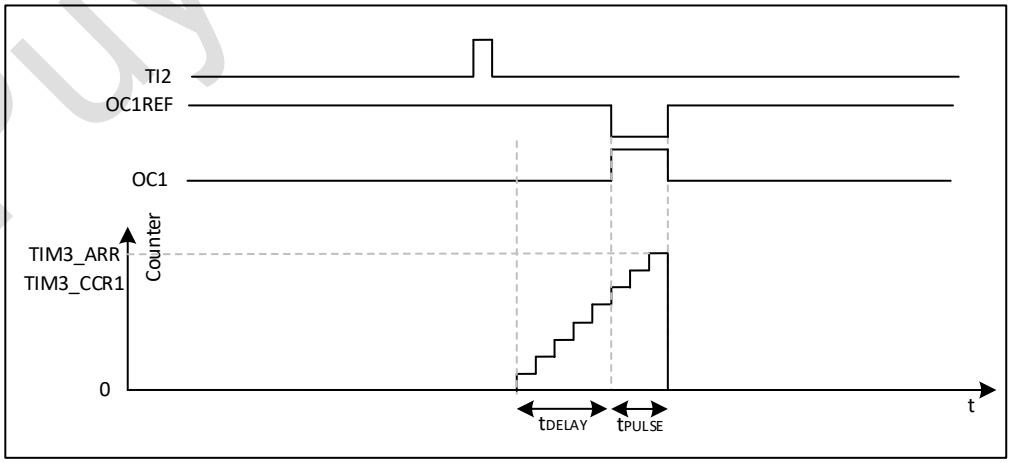


Figure 14-34 Example of single pulse mode

For example, you need to generate a positive pulse of length tPULSE on OC1 after a delay of tDELAY starting from the detection of a rising edge on the TI2 input pin.

Assuming TI2FP2 as the trigger.

- Set CC2S=01 in the TIMx_CMR1 register to map TI2FP2 to TI2.
- Set CC2P=0 in the TIMx_CCER register to enable the TI2FP2 to detect rising edges.
- Set TS=110 in the TIMx_SMCR register, TI2FP2 triggers as a slave mode controller (TRGI).
- Set SMS=110 in the TIMx_SMCR register (trigger mode), the TI2FP2 is used to start the counter.

The OPM waveform is determined by the value written to the comparison register (taking into account the clock frequency and the counter prescaler)

- tDELAY is defined by the value in the TIMx_CCR1 register.
- tPULSE is defined by the difference between the auto-load value and the compare value (TIMx_ARR - TIMx_CCR1).
- Assuming that a waveform from 0 to 1 is to be generated when a comparison match occurs, and a waveform from 1 to 0 is to be generated when the counter reaches the preload value; first set OC1M=111 in the TIMx_CMR1 register to enter PWM mode 2; selectively enable the preload registers as needed: set OC1PE=1 in TIMx_CMR1 and TIMx_CRM=1 in TIMx_CRM1 to enter PWM mode 2; set OC1PE=1 in TIMx_CMR1 and TIMx_CRM=1 in TIMx_CRM1 to enter PWM mode 2. 1 in TIMx_CCMR1 and ARPE in TIMx_CR1; then fill in the comparison value in TIMx_CCR1 register and the auto-load value in TIMx_ARR register, set the UG bit to generate an update event, and wait for an external trigger event on TI2. In this example, CC1P = 0.

In this example, the DIR and CMS bits in the TIMx_CR1 register should be set low.

Since only one pulse is required, OPM=1 in the TIMx_CR1 register must be set to stop counting at the next update event (when the counter flips from the auto-load value to 0). When OPM=0, repeat mode is selected.

14.2.11.1. Special case: OCx fast enable:

In single pulse mode, the CEN bit is set by the edge detection logic at the TIx input pin to start the counter. Comparison operations between the counter and the comparison value then produce the conversion of the output. However, these operations require a certain number of clock cycles, so it limits the minimum delay tDELAY that can be obtained.

To output a waveform with minimum delay, the OCxFE bit in the TIMx_CMRx register can be set; at this point OCxREF (and OCx) responds directly to the excitation and no longer relies on the result of the comparison, and the output waveform is the same as it would have been if the comparisons had been matched. OCxFE only works when the channel is configured for PWM1 and PWM2 modes.

14.2.12. Encoder interface mode

The encoder interface mode is selected by setting SMS=001 in the TIMx_SMCR register if the counter only counts on the TI2 edge, SMS=010 if it only counts on the TI1 edge, or SMS=011 if the counter counts on both the TI1 and TI2 edges.

TI1 and TI2 polarity can be selected by setting the CC1P and CC2P bits in the TIMx_CCER register; the input filters can also be programmed if required.

The two inputs TI1 and TI2 are used as interfaces to the incremental encoder. Assuming that the counter has been started (CEN=1 in the TIMx_CR1 register), the counter is driven by each valid trip on TI1FP1 or TI2FP2.TI1FP1 and TI2FP2 are the signals of TI1 and TI2 after passing them through the input filters and the polarity control; if there is no filtering and no phasing, then TI1FP1 = TI1 and TI2FP2 = The counting pulses and direction signals are generated according to the hopping sequence of the two input signals. Depending on the hopping sequence of the two input signals, the counter counts up or down while the hardware sets the DIR bit of the TIMx_CR1 register accordingly. Whether the counter counts on TI1, on TI2, or on both TI1 and TI2, a trip on either input (TI1 or TI2) recalculates the DIR bit.

The encoder interface mode is basically equivalent to using an external clock with direction selection. This means that the counter only counts continuously between 0 and the auto-loaded value of the TIMx_ARR register (either 0 to ARR counting or ARR to 0 counting, depending on the direction). So TIMx_ARR must be configured before counting starts; again, the capture, comparator, prescaler, trigger output characteristics, etc. still work as usual. The encoder mode and the external clock mode 2 are not compatible and therefore cannot be operated at the same time. In this mode, the counter is automatically modified according to the speed and direction of the incremental encoder, so that the contents of the counter always indicate the position of the encoder. The counting direction corresponds to the direction of rotation of the connected sensor. The following table shows all possible combinations, assuming that TI1 and TI2 do not change at the same time.

Table 14-1 Count Direction vs. Encoder Signal

Effective Edge	Relative signal levels (TI2 for TI1FP1 and TI1 for TI2FP2)	TI1FP1 signal		TI2FP2 signal	
		Up	Down	Up	Down
Count on TI1 only	High	Count Down	Count Up	Not count	Not count
	Low	Count Up	Count Down	Not Count	Not Count
Count on TI2 only	High	Not count	Not count	Count Up	Count Down
	Low	Not count	Not count	Count Down	Count Up
Counts on TI1 and TI2	High	Count Down	Count Up	Count Up	Count Down
	Low	Count Up	Count Down	Count Down	Count Up

An external incremental encoder can be connected directly to the MCU without the need for external interface logic. However, a comparator is typically used to convert the differential output of the encoder to a digital signal, which greatly increases the immunity to noise interference. The third signal from the encoder output represents a mechanical zero, which can be connected to an external interrupt input and trigger a counter reset.

The figure below is an example of counter operation, showing the generation of the counting signal and direction control. It also shows how input jitter is suppressed when a double edge is selected; jitter may be generated when the sensor's position is close to a transition point. In this example, we assume the following configuration:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped to IC1)
- CC2S='01' (TIMx_CCMR2 register, TI2FP2 mapped to IC2)

- CC1P='0' (TIMx_CCER register, TI1FP1 not inverted, TI1FP1=TI1)
- CC2P='0' (TIMx_CCER register, TI2FP2 not inverted, TI2FP2=TI2)
- SMS='011' (TIMx_SMCR register, all inputs are valid on rising and falling edges).
- CEN='1' (TIMx_CR1 register, counter enable).

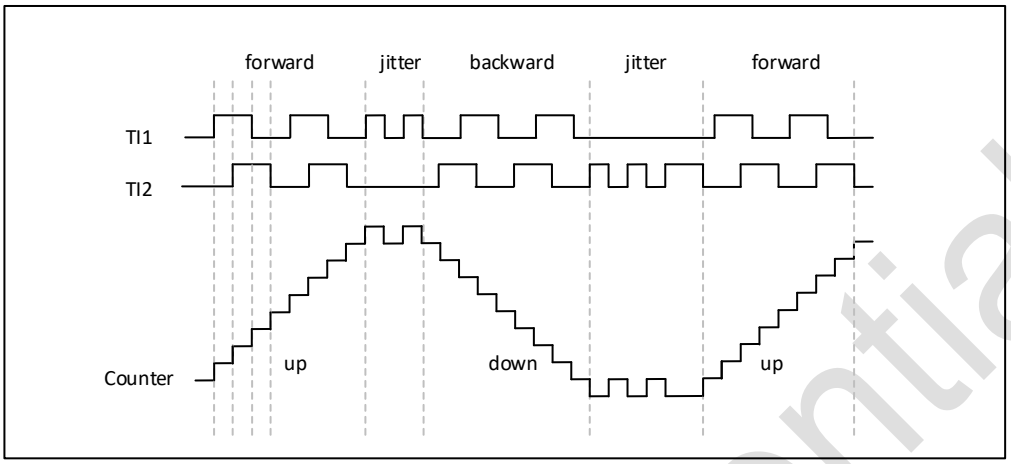


Figure 14-35 Example of Counter Operation in Encoder Mode

The following figure shows an example of counter operation when IC1FP1 polarity is inverted (CC1P='1', other configurations are the same as the above example)

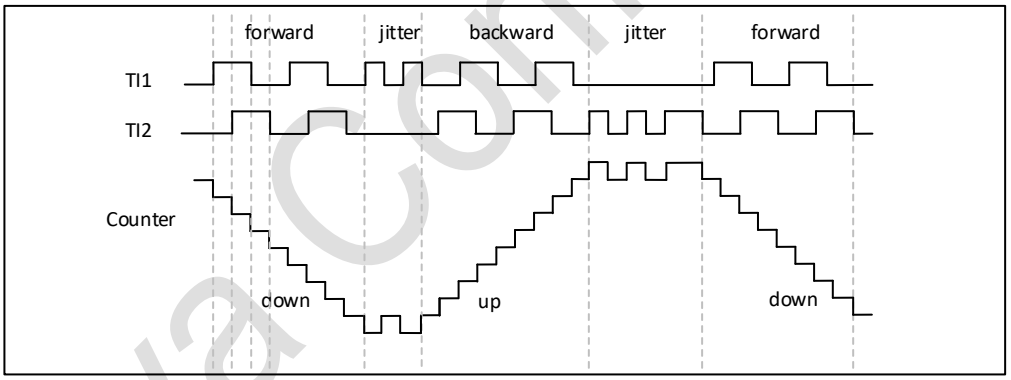


Figure 14-36 Example of counter operation when IC1FP1 polarity is inverted in encoder mode

When the timer is configured in encoder interface mode, it provides information about the current position of the sensor. By configuring the second timer in capture mode, the interval between two encoder events can be measured to obtain information about the dynamics (velocity, acceleration, deceleration). The encoder output indicating the mechanical zero point can be used for this purpose. Depending on the interval between the two events, the counter can be read out at a fixed time. If possible, you can latch the value of the counter to a third input capture register (the capture signal must be periodic and can be generated by another timer); or you can read its value through a DMA request generated by the real-time clock.

14.2.13. Timer input heterodyne function

The TI1S bit in the TIMx_CR2 register allows the input filters of channel 1 to be connected to the outputs of a heterodyne gate whose three inputs are TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The hetero-or outputs can be used for all timer input functions such as triggering or input capture. TIM1_8 SPEC gives an example of this feature being used to connect a Hall sensor.

14.2.14. Synchronisation of TIMx timers and external triggers

The TIMx timer can be synchronised with an external trigger in several modes: reset mode, gated mode and triggered mode.

14.2.14.1. Slave mode: reset mode

On the occurrence of a triggered input event, the counter and its prescaler can be reinitialised; at the same time an update event UEV is generated if the UDIS bit of the TIMx_CR1 register is low; all preloaded registers (TIMx_ARR, TIMx_CCRx) are then updated.

In the following example, a rising edge on the TI1 input causes the up counter to be cleared to zero:

- Configure channel 1 to detect the rising edge of TI1. Configure the bandwidth of the input filter (in this example, no filter is required, so keep IC1F=0000). The capture prescaler is not used in the trigger operation, so no configuration is required. the CC1S bit selects the input capture source only, i.e., CC1S=01 in the TIMx_CMR1 register. set CC1P=0 in the TIMx_CCER register to determine the polarity (detects only the rising edge).
- Set SMS=100 in the TIMx_SMCR register to configure the timer for reset mode; set TS=101 in the TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in TIMx_CR1 register to start the counter.

The counter starts counting according to the internal clock, and then operates normally until a rising edge appears in TI1; at this time, the counter is cleared to zero and then restarts counting from zero. At the same time, the trigger flag (TIF bit in the TIMx_SR register) is set, generating either an interrupt request or a DMA request depending on the setting of the TIE (interrupt enable) bit and the TDE (DMA enable) bit in the TIMx_DIER register.

The following figure shows the action when the auto-reload register TIMx_ARR = 0x36. The delay between the rising edge of TI1 and the actual reset of the counter depends on the resynchronisation circuitry at the TI1 input.

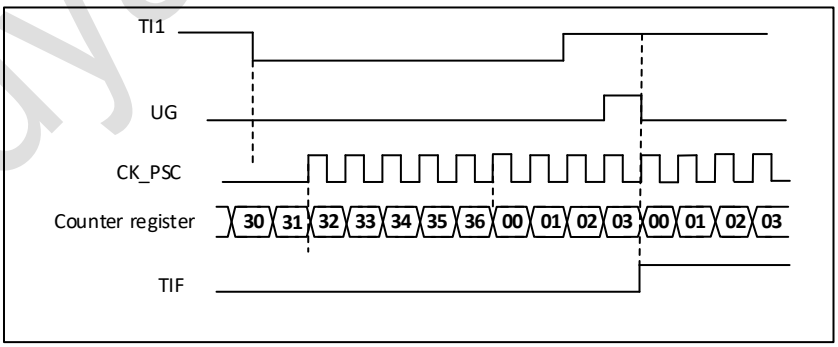


Figure 14-37 Control Circuit in Reset Mode

14.2.14.2. Slave mode: gated mode

Enable the counter according to the level of the selected input.

In the following example, the counter only counts up when TI1 is low:

- Configure channel 1 to detect a low level on TI1. Configure the input filter bandwidth (in this example, no filtering is required, so keep IC1F=0000). The capture prescaler is not used in the trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source, set CC1S=01 in the TIMx_CMR1 register. set CC1P=1 in the TIMx_CCER register to determine the polarity (detects only low levels).
- Set SMS=101 in the TIMx_SMCR register to configure the timer for gated mode; set TS=101 in the TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in TIMx_CR1 register to start the counter. In gated mode, if CEN=0, the counter cannot start, regardless of the trigger input level.

As long as TI1 is low, the counter starts counting based on the internal clock and stops counting once TI1 goes high. The TIF flag in TIMx_SR is set when the counter starts or stops.

The delay between the rising edge of TI1 and the actual stopping of the counter depends on the resynchronisation circuit at the TI1 input.

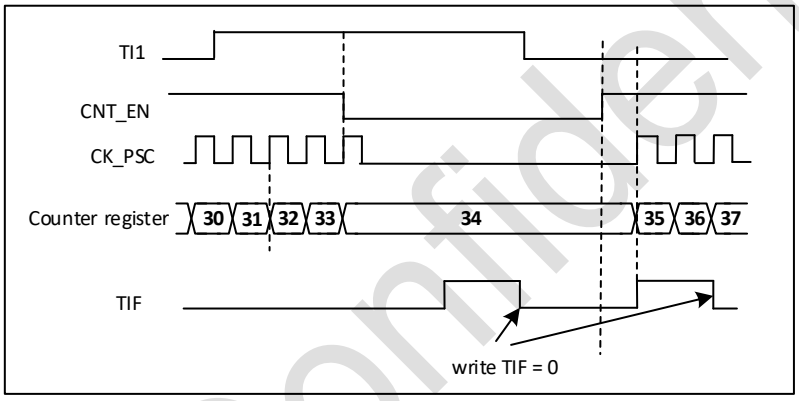


Figure 14-38 Control Circuit in Gated Mode

14.2.14.3. Slave mode: trigger mode

The selected event on the input enables the counter.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

- Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is required, keeping IC2F=0000). The capture prescaler is not used in the trigger operation and does not need to be configured. the CC2S bit is only used to select the input capture source, set CC2S=01 in the TIMx_CMR1 register. set CC2P=1 in the TIMx_CCER register to determine the polarity (detects only low levels).
- Set SMS=110 in the TIMx_SMCR register to configure the timer for trigger mode; set TS=110 in the TIMx_SMCR register to select TI2 as the input source.

When a rising edge of TI2 occurs, the counter starts counting driven by the internal clock and the TIF flag is set at the same time.

The delay between the rising edge of TI2 and the counter starting to count depends on the resynchronisation circuit at the TI2 input.

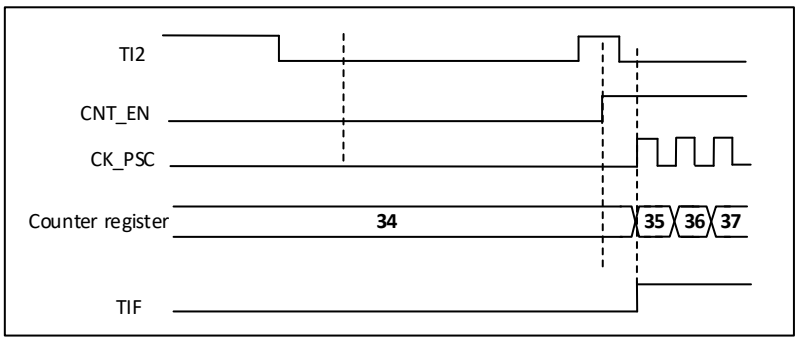


Figure 14-39 Control Circuit in Trigger Mode

14.2.14.4. Slave mode: external clock mode 2 + trigger mode

External clock mode 2 can be used with another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as an input to the external clock, and another input can be selected as a trigger input in reset mode, gated mode or trigger mode. It is not recommended to use the TS bit of the TIMx_SMCR register to select ETR as the TRGI.

In the following example, once a rising edge occurs on TI1, the counter counts up once on each rising edge of ETR:

- Configure the external trigger input circuit via the TIMx_SMCR register:

– ETF=0000: no filtering

ETPS=00: no prescaler.

– ETP=0: detect the rising edge of ETR, set ECE=1 to enable external clock mode 2.

- Configure channel 1 as follows to detect the rising edge of TI:

– IC1F=0000: no filtering

– Capture prescaler is not used in the trigger operation, no configuration is required.

– Set CC1S=01 in TIMx_CMR1 register to select input capture source.

Set CC1P=0 in TIMx_CCER register to determine polarity (detect rising edge only).

- Set SMS=110 in the TIMx_SMCR register to configure the timer for trigger mode. Set TS=101 in the TIMx_SMCR register to select TI1 as the input source.

When a rising edge occurs on TI1, the TIF flag is set and the counter starts counting on the rising edge of ETR.

The delay between the rising edge of the ETR signal and the actual reset of the counter depends on the resynchronisation circuit at the ETRP input.

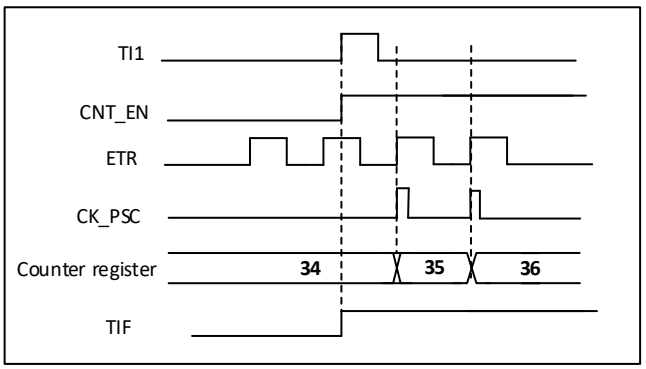


Figure 14-40 Control circuit in external clock mode 2 + trigger mode

14.2.15. Timer Synchronisation

All TIMx timers are connected internally for timer synchronisation or linking. When one timer is in master mode, it can reset, start, stop, or provide a clock to the counter of another timer that is in slave mode.

The following figure shows an overview of the Trigger Select and Master Mode Select modules.

14.2.15.1. Use one timer as a prescaler for another timer

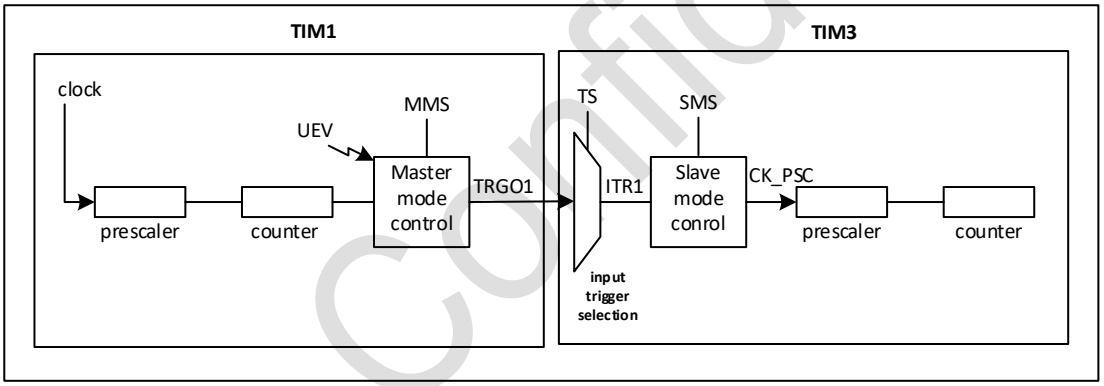


Figure 14-41 Example of a Master/Slave Timer

For example, you can configure Timer 1 as a prescaler for Timer 2. Perform the following operations:

● Configure Timer 1 as the master mode, which can output a periodic trigger signal at each update event UEV. With MMS='010' in the TIM1_CR2 register, output a rising edge signal on TRGO1 whenever an update event is generated.

● Connect the TRGO1 output of Timer1 to Timer2, set TS='000' of the TIM2_SMCR register, and configure Timer2 to be in slave mode using ITR1 as the internal trigger.

● The slave mode controller is then placed in external clock mode 1 (SMS=111 of the TIM2_SMCR register); this allows Timer 2 to be driven by the periodic rising edge (i.e. Timer 1's counter overflow) signal from Timer 1.

● Finally, the CEN bit of the corresponding (TIMx_CR1 register) must be set to start each of the two timers.

Note: If OCx has been selected as the trigger output for Timer 1 (MMS=1xx), its rising edge is used to drive Timer 2's counter.

14.2.15.2. Use a timer to enable another timer

In this example, the enable of Timer 2 is controlled by the output comparison of Timer 1.

Timer 2 counts the divided internal clock only when OC1REF of Timer 1 is high. The clock frequency of both timers is obtained by dividing CK_INT by 3 (fCK_CNT = fCK_INT/3) by the prescaler.

- Configure Timer 1 to be the master mode and send its output comparison reference signal (OC1REF) as the trigger output (MMS=100 in TIM1_CR2 register)
- Configure the OC1REF waveform of Timer 1 (TIM1_CMR1 register)
- Configure Timer 2 to get input triggering from Timer 1 (TS=000 of TIM2_SMCR register)
- Configure Timer 2 for gated mode (SMS=101 of TIM2_SMCR register)
- Set CEN=1 of the TIM2_CR1 register to enable Timer2
- Set CEN=1 of TIM1_CR1 register to enable Timer1.

Note: The clock of Timer 2 is not synchronised with the clock of Timer 1. This mode only affects the enable signal of the Timer 2 counter.

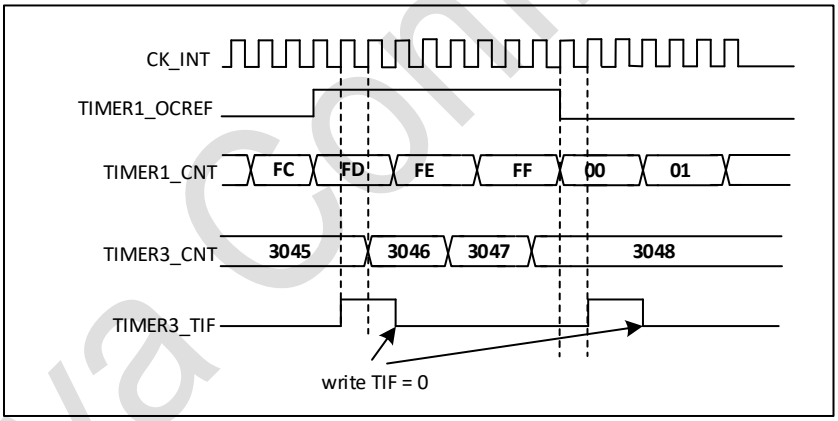


Figure 14-42 OC1REF for Timer 1 Controls Timer 2

In the example of Figure, before Timer 2 is started, their counters and prescalers are not initialised, so they start counting from the current value. It is possible to reset the 2 timers before starting Timer 1 so that they start from a given value, i.e., write any value needed in the timer counter. The timers can be reset by writing the UG bit of the TIMx_EGR register.

In the next example, it is necessary to synchronise Timer 1 and Timer 2. Timer 1 is in master mode and starts from 0, Timer 2 is in slave mode and starts from 0xE7; the prescaler coefficients are the same for both timers. Writing '0' to the CEN bit of TIM1_CR1 will disable Timer 1 and Timer 2 will then stop.

● Configure Timer 1 to be in master mode and send the Output Compare 1 reference signal (OC1REF) as the trigger output (MMS=100 in TIM1_CR2 register).

● Configure the OC1REF waveform for Timer 1 (TIM1_CMR1 register).

● Configure Timer 2 to get input triggering from Timer 1 (TS=000 in the TIM2_SMCR register).

● Configure Timer 2 for gated mode (SMS=101 of the TIM2_SMCR register).

● Set UG='1' in the TIM1_EGR register to reset Timer 1.

● Set UG='1' of TIM2_EGR register to reset Timer 2.

● Write '0XE7' to Timer 2's counter (TIM2_CNTL) to initialise it to 0xE7.

● Set CEN='1' in the TIM2_CR1 register to enable Timer 2.

● Set CEN='1' of TIM1_CR1 register to enable Timer 1.

● Set CEN='0' of TIM1_CR1 register to stop Timer 1.

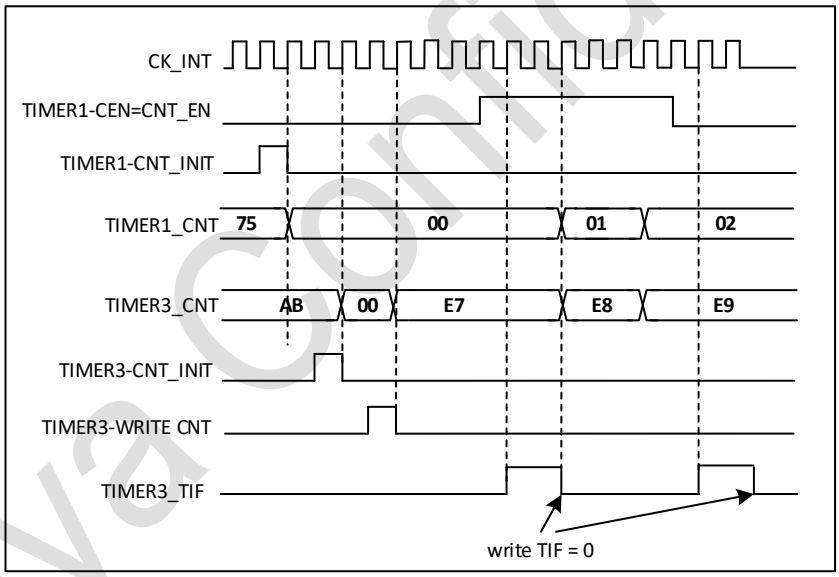


Figure 14-43 Timer 2 can be controlled by enabling Timer 1

14.2.15.3. Use a timer to start another timer

In this example, Timer 2 is enabled using an update event from Timer 1. As soon as Timer 1 generates an update event, Timer 2 starts counting from its current value (which can be non-zero) according to the divided internal clock. On receipt of a trigger signal, the CEN bit of Timer 2 is automatically set to '1' and the counter starts counting until a '0' is written to the CEN bit of the TIM2_CR1 register. Both timers are clocked by dividing the prescaler pair CK_INT by 3 (fCK_CNT = fCK_INT/3).

● Configure Timer 1 to be in master mode, sending its update event (UEV) as a trigger output (MMS=010 in the TIM1_CR2 register).

- Configure the period of Timer 1 (TIM1_ARR register).
- Configure Timer 2 to get input triggering from Timer 1 (TS=000 of the TIM2_SMCR register).
- Configure Timer 2 for trigger mode (SMS=110 of the TIM2_SMCR register).
- Set CEN=1 of the TIM1_CR1 register to start Timer 1.

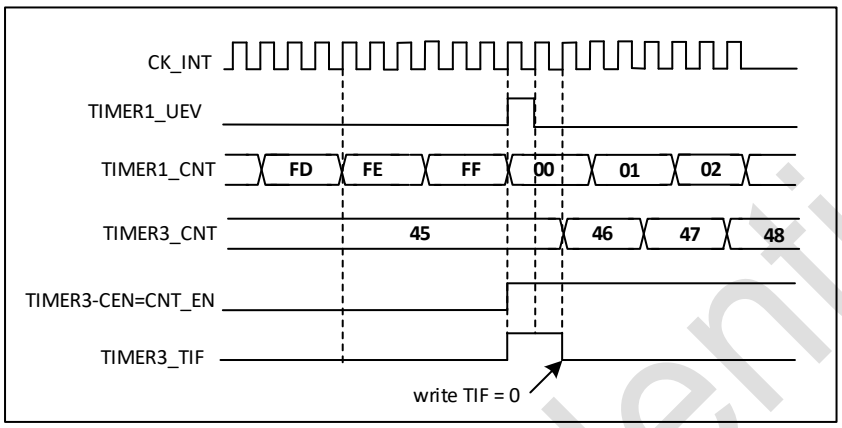


Figure 14-44 Triggering Timer 2 Using Timer 1 Updates

In the previous example, it is possible to initialise both counters before starting the count. Shows the action in the same configuration as 0, using trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

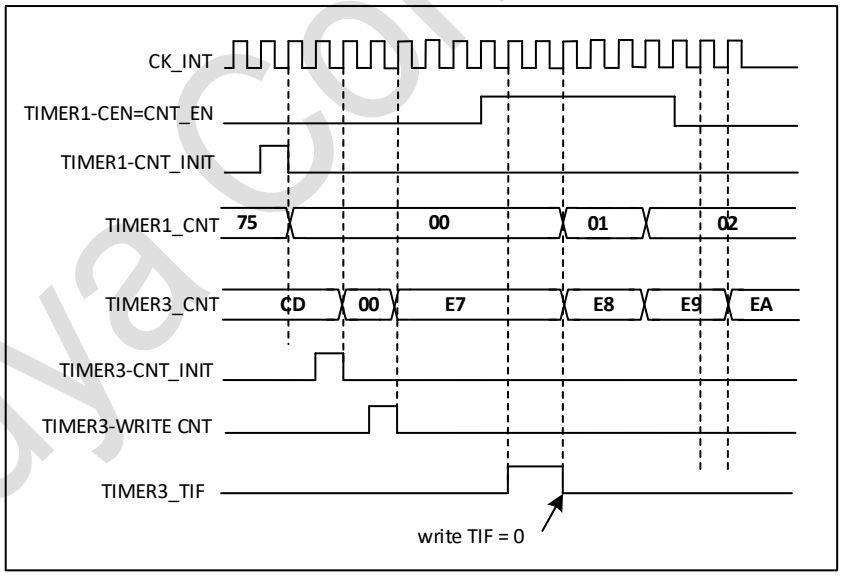


Figure 14-45 Triggering Timer 2 Using Timer 1 Enable

14.2.15.4. Use a timer as a prescaler for another one

This example uses Timer 1 as a prescaler for Timer 2. Configure as follows:

- Configure Timer 1 to be in master mode, sending its update event UEV as a trigger output (MMS='010' in the TIM1_CR2 register). Then output a cycle signal each time the counter overflows;
- Configure the period of Timer 1 (TIM1_ARR register);

- Configure Timer 2 to get an input trigger from Timer 1 (TS=000 of the TIM2_SMCR register);
- Configure Timer 2 to use external clock mode (SMS=111 of the TIM2_SMCR register);
- Set CEN=1 of the TIM1_CR2 register to start Timer 2;
- Set CEN=1 of the TIM1_CR1 register to start Timer1.

14.2.15.5. Activate 2 timers synchronously using an external trigger

This example enables Timer 1 when the TI1 input to Timer 1 rises, and enables Timer 2 at the same time as Timer 1. see Figure 4-48. to ensure counter alignment, Timer 1 must be configured in master/slave mode (corresponding to TI1 as slave, corresponding to Timer 2 as master):

- Configure Timer 1 to be in master mode, sending its enable as a trigger output (MMS=001 in the TIM1_CR2 register).
- Configure Timer1 as slave mode, get input trigger from TI1 (TS=100 of TIM1_SMCR register).
- Configure Timer 1 for trigger mode (SMS=110 of TIM1_SMCR register).
- Configure Timer 1 for master/slave mode with MSM=1 in the TIM1_SMCR register.
- Configure Timer 2 to get input trigger from Timer 1 (TS=000 of TIM2_SMCR register).
- Configure Timer 2 for trigger mode (SMS=110 of TIM2_SMCR register).

When a rising edge occurs on TI1 of Timer 1, both timers synchronously start counting according to the internal clock and both TIF flags are set simultaneously.

Note: In this example, both timers are initialised (set the corresponding UG bit) before start-up and both counters start from 0, but an offset can be inserted between the timers by writing to any of the counter registers (TIMx_CNT). In the figure below you can see that in master/slave mode there is a delay between CNT_EN and CK_PSC in timer 1.

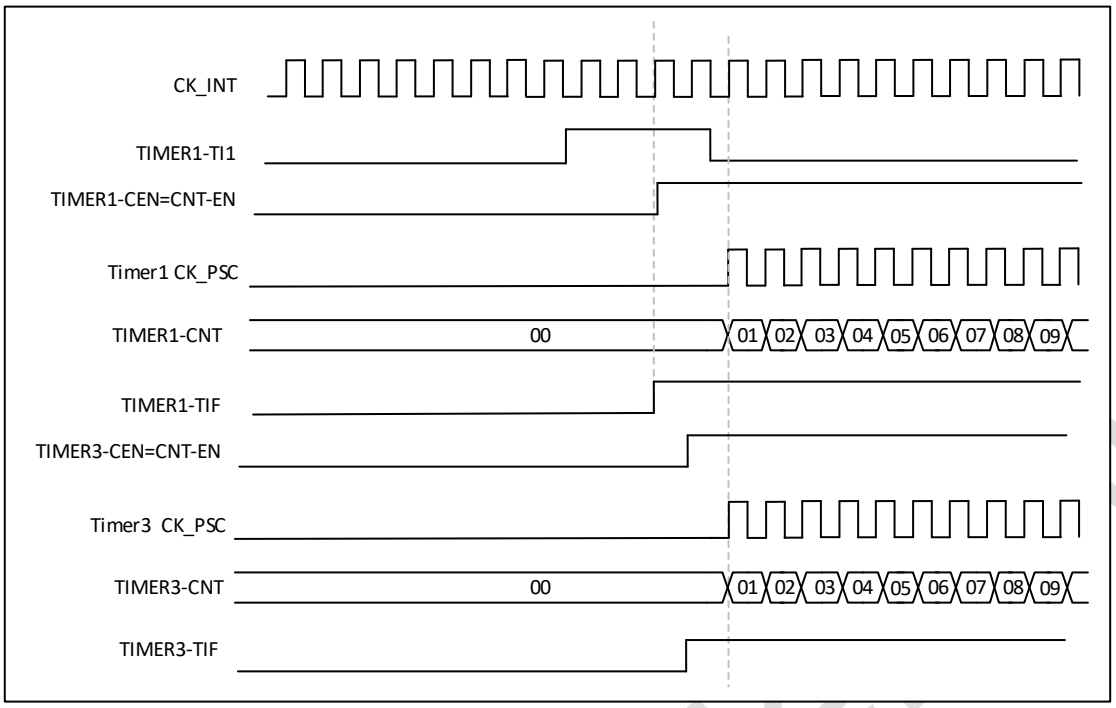


Figure 14-46 Triggering Timer 1 and Timer 2 Using the TI1 Input of Timer 1

14.2.16. Debug Mode

When the microcontroller enters debug mode (Cortex-M4 core stop), depending on the DBG_TIMx_STOP setting in the DBG module, the

TIMx counter can either continue normal operation or stop.

14.3. Register Description

TIM2 register base address: 0x4000 0000

TIM3 register base address: 0x4000 0400

TIM4 register base address: 0x4000 0800

TIM5 register base address: 0x4000 0C00

14.3.1. TIM2/3/4/5 control register 1 (TIMx_CR1)

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
Reserved						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:10	Reserved, always 0			
9:8	CKD	RW	0	Clock division These two bits define the ratio of the division between the frequency of the timer clock (CK_INT) and the sampling clock (tDTS) used by the digital filter (ETR, TIx). 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: Reserved, do not use this configuration
7	ARPE	RW	0	Auto-reload preload enable bit 0: TIMx_ARR register is not buffered; 1: TIMx_ARR register is buffered.
6:5	CMS	RW	0	Center-aligned mode selection 00: Edge-aligned mode. The counter counts up or down according to the direction bit (DIR). 01: Centre-aligned mode 1. the counter alternately counts up and down. The output compare interrupt flag bit of the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set only when the counter is counting down. 10: Central Alignment Mode 2. the counter alternately counts up and down. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set only when the counter counts up. 11: Central Alignment Mode 3. the counter alternately counts up and down. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set both when the counter counts up and down. Note: While the counter is on (CEN=1), the transition from edge-aligned mode to centre-aligned mode is not allowed.
4	DIR	RW	0	Direction 0: Counter counts up; 1: Counter counts down. Note: This bit is read-only when the counter is configured for centre-aligned mode or encoder mode.
3	OPM	RW	0	One pulse mode 0: The counter does not stop when an update event occurs; 1: The counter stops when the next update event occurs (CEN bit is cleared).
2	URS	RW	0	Update request source Software selects the source of UEV events with this bit. 0: If an update interrupt or DMA request is enabled, either of the following events generates an update interrupt or DMA request: - Counter overflow/underflow - Setting of the UG bit - Update generated from the mode controller

Bit	Name	R/W	Reset Value	Function
				1: If an update interrupt or DMA request is enabled, only a counter overflow/underflow generates an update interrupt or DMA request.
1	UDIS	RW	0	Update disable The software allows/disables the generation of UEV events with this bit 0: UEV is allowed. update (UEV) events are generated by any of the following events: - Counter overflow/underflow - Setting of the UG bit - Update generated from the mode controller Registers with caches are loaded with their preloaded values. (Translation: update of shadow registers) 1: Disable UEV. no update event is generated and the shadow registers (ARR, PSC, CCRx) keep their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescalers are reinitialised.
0	CEN	RW	0	Counter enable 0: Disable the counter; 1: Enables the counter. Note: The external clock, gated mode and encoder mode can only work after the CEN bit is set in software. Trigger mode can automatically set the CEN bit in hardware.

14.3.2. TIM2/3/4/5 control registers2 (TIMx_CR2)

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
Reserved								RW	RW			RW	Reserved		

Bit	Name	R/W	Reset Value	Function
15:8	Reserved, always 0			
7	TI1S	RW	0	TI1 selection 0: The TIMx_CH1 pin is connected to the TI1 input; 1: TIMx_CH1, TIMx_CH2, and TIMx_CH3 pins are connected to the TI1 input after heterodyne.
6:4	MMS	RW	0	Master mode selection These 3 bits are used to select the synchronisation information (TRGO) to be sent to the slave timer in master mode. The possible combinations are as follows:

Bit	Name	R/W	Reset Value	Function
				000: Reset - The UG bit of the TIMx_EGR register is used as the trigger output (TRGO). If the reset is generated by a trigger input (from the mode controller being in reset mode), the signal on the TRGO will have a delay relative to the actual reset. 001: Enable - The counter enable signal CNT_EN is used as the trigger output (TRGO). Sometimes it is necessary to start more than one timer at the same time or to control the enabling of a slave timer over a period of time. The counter enable signal is generated by a logical or of the CEN control bit and the trigger input signal in gated mode. When the counter enable signal is controlled by a trigger input, there is a delay on TRGO unless master/slave mode is selected (see description of the MSM bit in the TIMx_SMCR register). 010: Update - The update event is selected as the trigger input (TRGO). For example, a master timer clock can be used as a prescaler for a slave timer. 011: Compare Pulse - The trigger output sends a positive pulse (TRGO) when the CC1IF flag is to be set (even if it is already high) when a capture or a successful comparison occurs. 100: The Compare - OC1REF signal is used as the trigger output (TRGO). 101: Compare - OC2REF signal is used as trigger output (TRGO). 110: Compare - OC3REF signal is used as trigger output (TRGO). 111: Compare - OC4REF signal is used as trigger output (TRGO). NOTE: The slave timer and ADC clocks must first be enabled to receive signals from the master timer and should not be altered on reception.
3	CCDS	RW	0	Capture/compare DMA selection 0: DMA request for CCx is sent when a CCx event occurs; 1: DMA request for CCx is sent when an update event occurs.
2:0	Reserved, always reads 0.			

14.3.3. TIM2/3/4/5 slave mode control registers (TIMx_SMCR)

Address offset:0x08

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
RW	RW	RW		RW				RW	RW			Reserved	RW		

Bit	Name	R/W	Reset Value	Function
15	ETP	RW	0	External trigger polarity This bit selects whether to use ETR or the inverse of ETR as the trigger operation. 0: ETR is not inverted, active high or rising edge; 1: ETR is inverted, active low or falling edge.
14	ECE	RW	0	External clock enable bit This bit enables external clock mode 2 0: disables external clock mode 2; 1: enables external clock mode 2. the counter is driven by any valid edge on the ETRF signal. Note 1: Setting the ECE bit has the same effect as selecting external clock mode 1 and connecting TRGI to ETRF (SMS=111 and TS=111). Note 2: The following slave modes can be used in conjunction with External Clock Mode 2: Reset Mode, Gated Mode, and Trigger Mode; however, the TRGI cannot be connected to the ETRF at this time (the TS bit cannot be '111'). Note 3: When External Clock Mode 1 and External Clock Mode 2 are enabled at the same time, the input of the external clock is ETRF. Note 4: ch1 and etr of tim2 share the same io port and should be avoided at the same time.
13:12	ETPS	RW	0	External trigger prescaler The frequency of the external trigger signal ETRP must be at most 1/4 of the TIMxCLK frequency. prescaler can be used to reduce the frequency of ETRP when a faster external clock is input. 00: Turns off the pre-divided frequency; 01: ETRP frequency divided by 2; 10: ETRP frequency divided by 4; 11: ETRP frequency divided by 8.
11:8	ETF	RW	0	External trigger filter These bits define the frequency at which the ETRP signal is sampled and the bandwidth at which the ETRP is digitally filtered. In effect, the digital filter is an event counter that records N events and then generates a jump in the output. 0000: no filter, sampled at fDTS 0001: Sampling frequency fSAMPLING=fCK_INT, N=2 0010: Sampling frequency fSAMPLING=fCK_INT, N=4 0011: Sampling frequency fSAMPLING=fCK_INT, N=8 0100: Sampling frequency fSAMPLING=fDTS/2, N=6 0101: Sampling frequency fSAMPLING=fDTS/2, N=8 0110: Sampling frequency fSAMPLING=fDTS/4, N=6 0111: Sampling frequency fSAMPLING=fDTS/4, N=8 1000: Sampling frequency fSAMPLING=fDTS/8, N=6 1001: Sampling frequency fSAMPLING=fDTS/8, N=8 1010: Sampling frequency fSAMPLING=fDTS/16, N=5 1011: Sampling frequency fSAMPLING=fDTS/16, N=6 1100: Sampling frequency fSAMPLING=fDTS/16, N=8

Bit	Name	R/W	Reset Value	Function
				1101: Sampling frequency fSAMPLING=fDTS/32, N=5 1110: Sampling frequency fSAMPLING=fDTS/32, N=6 1111: Sampling frequency fSAMPLING=fDTS/32, N=8
7	MSM	RW	0	Master/slave mode 0: No effect; 1: The event on the trigger input (TRGI) is delayed to allow perfect synchronisation between the current timer (via TRGO) and its slave timer. This is useful when it is required to synchronise several timers to a single external event.
6:4	TS	RW	0	Trigger selection This 3-bit selection is used to synchronise the trigger input of the counter. 000: Internal trigger 0 (ITR0) 100: Edge detector of TI1 (TI1F_ED) 001: Internal trigger 1 (ITR1) 101: Filtered timer input 1 (TI1FP1) 010: Internal trigger 2 (ITR2) 110: Filtered timer input 2 (TI2FP2) 011: Internal trigger 3 (ITR3) 111: External trigger input (ETRF) For more details on ITRx, see Table 5-1. Note: These bits can only be changed when not used (e.g. SMS=000) to avoid false edge detection when changed.
3	Reserved, always reads 0.			
2:0	SMS	RW	0	Slave mode selection When an external signal is selected, the active edge of the trigger signal (TRGI) is related to the polarity of the selected external input (see description of input control registers and control registers) 000: Off Slave Mode - If CEN=1, the prescaler is driven directly from the internal clock. 001: Encoder Mode 1 - Depending on the level of TI1FP1, the counter counts up/down on the edge of TI2FP2. 010: Encoder Mode 2 - Depending on the level of TI2FP2, the counter counts up/down on the edge of TI1FP1. 011: Encoder Mode 3 - Depending on the input level of another signal, the counter counts up/down on the edges of TI1FP1 and TI2FP2. 100: Reset Mode - The rising edge of the selected trigger input (TRGI) reinitialises the counter and generates a signal to update the registers. 101: Gated Mode - When the trigger input (TRGI) is high, the counter is clocked on. Once the trigger input goes low, the counter is stopped (but not reset). Counter start and stop are controlled. 110: Trigger Mode - The counter is started (but not reset) on the rising edge of the trigger input TRGI, only the start of the counter is controlled. 111: External Clock Mode 1 - The rising edge of the selected trigger input (TRGI) drives the counter. Note: Do not use gated mode if TI1F_EN is selected as the trigger input (TS=100). This is because, TI1F_ED outputs a pulse each time TI1F changes, however the gated mode is to check the level of the trigger input. Note: Do not use uev as the trgo output signal in encoder mode, (i.e. mms cannot be configured to 010)

Table 14-2 TIMx internal trigger connection

Slave Timer	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM2	TIM1	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

14.3.4. TIM2/3/4/5 DMA/Interrupt Enable Registers (TIMx_DIER)

Address offset:0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TDE	Reserved	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE
Reserved	RW	Reserved	RW	RW	RW	RW	RW	Reserved	RW	Reserved	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	Reserved, always 0			
14	TDE	RW	0	Trigger DMA request enable 0: Trigger DMA request is disabled; 1: Trigger DMA request is allowed.
13	Reserved, always 0			
12	CC4DE	RW	0	Capture/Compare 4 DMA request enable 0: Capture/Compare 4 DMA request disabled; 1: Capture/Compare 4 DMA request enable.
11	CC3DE	RW	0	Capture/Compare 3 DMA request enable 0: Capture/Compare 3 DMA request disabled; 1: Capture/Compare 3 DMA request enable.
10	CC2DE	RW	0	Capture/Compare 2 DMA request enable 0: Capture/Compare 2 DMA request disabled; 1: Capture/Compare 2 DMA request enable.
9	CC1DE	RW	0	Capture/Compare 1 DMA request enable 0: Capture/Compare 1 DMA request disabled; 1: Capture/Compare 1 DMA request enable.
8	UDE	RW	0	Update DMA request enable 0: Update DMA request disabled; 1: Update DMA request enabled.

Bit	Name	R/W	Reset Value	Function
7	Reserved, always 0			
6	TIE	RW	0	Trigger interrupt enable 0: Trigger interrupt disabled; 1: enable trigger interrupt.
5	Reserved, always 0			
4	CC4IE	RW	0	Capture/Compare 4 interrupt enable 0: Capture/Compare 4 interrupt disabled; 1: Capture/Compare 4 interrupt enable.
3	CC3IE	RW	0	Capture/Compare 3 interrupt enable 0: Capture/Compare 3 interrupt disabled; 1: Capture/Compare 3 interrupt enable.
2	CC2IE	RW	0	Capture/Compare 2 interrupt enable 0: Capture/Compare 2 interrupt is disabled; 1: Capture/Compare 2 interrupt enable.
1	CC1IE	RW	0	Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt disabled; 1: Capture/Compare 1 interrupt enable
0	UIE	RW	0	Update interrupt enable 0: Update interrupt disabled; 1: update interrupt enabled.

14.3.5. TIM2/3/4/5 status registers (TIMx_SR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF
Reserved			RC_W0	RC_W0	RC_W0	RC_W0	Reserved		RC_W0	Reserved	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
15:13	Reserved, always 0			
12	CC4OF	RC_W0	0	Capture/Compare 4 overcapture flag See CC1OF description.

Bit	Name	R/W	Reset Value	Function
11	CC3OF	RC_W0	0	Capture/Compare 3 overcapture flag See CC1OF description.
10	CC2OF	RC_W0	0	Capture/Compare 2 overcapture flag See CC1OF description.
9	CC1OF	RC_W0	0	Capture/Compare 1 overcapture flag This flag can be set by hardware to 1 only if the corresponding channel is configured for input capture. writing 0 clears this bit. 0: No overcapture is generated; 1: The state of CC1IF is already '1' when the counter value is captured into the TIMx_CCR1 register.
8:7	Reserved, always reads 0			
6	TIF	RC_W0	0	Trigger interrupt flag This position is '1' by hardware when a trigger event occurs (a valid edge is detected on the TRGI input when the slave mode controller is in a mode other than gated mode, or either edge in gated mode). It is cleared '0' by software. 0: No trigger event is generated; 1: Trigger interrupt waiting for response.
5	Reserved, always reads 0			
4	CC4IF	RC_W0	0	Capture/Compare 4 interrupt flag Refer to CC1IF for description.
3	CC3IF	RC_W0	0	Capture/Compare 3 interrupt flag Refer to CC1IF for description.
2	CC2IF	RC_W0	0	Capture/Compare 2 interrupt flag Refer to CC1IF for description.
1	CC1IF	RC_W0	0	Capture/Compare 1 interrupt flag If channel CC1 is configured for output mode: This bit is set to 1 by hardware when the counter value matches the compare value, except in centre-symmetric mode (refer to the CMS bit in the TIMx_CR1 register). It is cleared '0' by software. 0: No match occurs; 1: The value of TIMx_CNT matches the value of TIMx_CCR1. When the content of TIMx_CCR1 is greater than the content of TIMx_APR, the CC1IF bit goes high under counter overflow in up or up/down counting modes, or counter underflow conditions in down counting mode if channel CC1 is configured for input mode: This bit is set '1' by hardware when a capture event occurs, it is cleared '0' by software or by reading TIMx_CCR1. 0: No input capture is generated;

Bit	Name	R/W	Reset Value	Function
				1: The counter value has been captured (copied) to TIMx_CCR1 (an edge of the same polarity as the one selected is detected on IC1).
0	UIF	RC_W0	0	Update interrupt flag This bit is set '1' by hardware when an update event is generated. It is cleared to '0' by software. 0: No update event generated; 1: update interrupt waiting for response. This bit is set '1' by hardware when the register is updated: - If TIMx_CR1 register UDIS = 0, when the repeat counter value overflows or underflows (update event is generated when repeat counter = 0). - If URS=0 and UDIS=0 of the TIMx_CR1 register, the update event is generated when UG=1 of the TIMx_EGR register is set, and when the counter CNT is reinitialised by software. - If URS=0, UDIS=0 of TIMx_CR1 register, when counter CNT is reinitialised by a trigger event. (Refer to TIM2/3/4/5 Slave Mode Control Register (TIMx_SMCR)).

14.3.6. TIM2/3/4/5 event generation registers (TIMx_EGR)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG
Reserved									W	Reserved	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
15:7	Reserved, always 0			
6	TG	W	0	Trigger generation This bit is set '1' by the software to generate a trigger event, which is automatically cleared '0' by the hardware. 0: No action; 1: TIF=1 in the TIMx_SR register generates the corresponding interrupt and DMA if they are turned on.
5	Reserved, always 0			
4	CC4G	W	0	Generate Capture/Compare 4 generation
3	CC3G	W	0	Refer to CC1G description.
2	CC2G	W	0	Generate Capture/Compare 3 generation.
1	CC1G	W	0	Capture/Compare 1 generation

Bit	Name	R/W	Reset Value	Function
				This bit is set '1' by software to generate a capture/compare event, and is automatically cleared '0' by hardware. 0: No action; 1: Generate a capture/compare event on channel CC1: If channel CC1 is configured as an output: Set CC1IF=1 to generate the corresponding interrupt and DMA if they are turned on. If channel CC1 is configured as an input: The current counter value is captured to the TIMx_CCR1 register; set CC1IF=1 to generate the corresponding interrupt and DMA if it is switched on. set CC1OF=1 if CC1IF is already 1.
0	UG	W	0	Update generation This bit is set to '1' by software and cleared to '0' automatically by hardware. 0: No action; 1: Re-initialise the counter and generate an update event. Note that the prescaler counter is also cleared '0' (but the prescaler coefficients remain unchanged). The counter is cleared '0' if in centre-symmetric mode or if DIR=0 (counting up); if DIR=1 (counting down) the counter takes the value of TIMx_ARR.

14.3.7. TIM2/3/4/5 Capture/comparison mode control register 1 (TIMx_CCMR1)

Address offset:0x18

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				ICIF[3:0]				ICIPSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

14.3.7.1. Output comparison mode

Bit	Name	R/W	Reset Value	Function
15	OC2CE	RW	0	Output Compare 2 clear enable
14:12	OC2M	RW	0	Output Compare 2 mode
11	OC2PE	RW	0	Output Compare 2 preload enable
10	OC2FE	RW	0	Output Compare 2 fast enable
9:8	CC2S	RW	0	Capture/Compare 2 selection.

Bit	Name	R/W	Reset Value	Function
				This bit defines the direction of the channel (input/output), and the selection of the input pin: 00: The CC2 channel is configured as an output; 01: CC2 channel is configured as input, IC2 is mapped on TI2; 10: CC2 channel is configured as input, IC2 is mapped on TI1; 11: CC2 channel is configured as an input and IC2 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).
7	OC1CE	RW	0	Output Compare 1 clear '0' enable 0: OC1REF is not affected by the ETRF input; 1: Clear OC1REF=0 once the ETRF input is detected high.
6:4	OC1M	RW	0	Output Compare 1 mode This 3-bit bit defines the action of the output reference signal OC1REF, which determines the value of OC1. OC1REF is active high, and the effective level of OC1 depends on the CC1P bit. 000: Freeze. Comparison between output comparison register TIMx_CCR1 and counter TIMx_CNT does not work for OC1REF; 001: Set channel 1 to an active level when matched. Forces OC1REF high when the value of counter TIMx_CNT is the same as capture/comparison register 1 (TIMx_CCR1). 010: Set channel 1 to invalid level when matched. Force OC1REF low when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 011: flip-flop. Flips the level of OC1REF when TIMx_CCR1 = TIMx_CNT. 100: Force to invalid level. Forces OC1REF to be low. 101: Force to valid level. Forces OC1REF to be high. 110: PWM Mode 1- In up count, channel 1 is valid level once TIMx_CNT<TIMx_CCR1, otherwise it is invalid level; in down count, channel 1 is invalid level once TIMx_CNT>TIMx_CCR1 (OC1REF=0), otherwise it is valid level (OC1REF= 1). 111: PWM Mode 2- In up count, channel 1 is invalid level once TIMx_CNT<TIMx_CCR1, otherwise it is valid level; in down count, channel 1 is valid level once TIMx_CNT>TIMx_CCR1, otherwise it is invalid level. Note 1: Once LOCK level is set to 3 (LOCK bit in TIMx_BDTR register) and CC1S=00 (the channel is configured as an output) then this bit cannot be modified. Note 2: In PWM mode 1 or PWM mode 2, the OC1REF level is changed only when the comparison result is changed or when switching from freeze mode to PWM mode in the output comparison mode.
3	OC1PE	RW	0	Output Compare 1 preload enable 0: Disable the preload function of TIMx_CCR1 register, TIMx_CCR1 register can be written at any time, and the newly written value takes effect immediately.

Bit	Name	R/W	Reset Value	Function
				1: Enable the preload function of TIMx_CCR1 register, read and write operations are only operated to the preloaded register, and the preloaded value of TIMx_CCR1 is loaded into the current register when the update event arrives. Note 1: This bit cannot be modified once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as an output). Note 2: Only in single pulse mode (OPM=1 in the TIMx_CR1 register), the PWM mode can be used without acknowledging the preload register, otherwise its action is uncertain.
2	OC1FE	RW	0	Output Compare 1 fast enable This bit is used to speed up the response of the CC outputs to trigger input events. 0: Depending on the value of the counter and CCR1, CC1 operates normally, even if the trigger is open. The minimum delay to activate the CC1 output is 5 clock cycles when the input to the trigger has a valid edge. 1: The active edge of the input to the flip-flop acts as if a compare match has occurred. Therefore, OC is set to the comparison level independent of the comparison result. The delay between the active edge of the sampling flip-flop and the CC1 output is reduced to 3 clock cycles. OCFE only functions when the channel is configured for PWM1 or PWM2 mode.
1:0	CC1S	RW	0	Capture/Compare 1 selection. These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: CC1 channel is configured as input, IC1 is mapped on TI2; 11: CC1 channel is configured as input and IC1 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC1S is writable only when the channel is off (CC1E=0 in the TIMx_CCER register).

14.3.7.2. Input capture mode

Bit	Name	R/W	Reset Value	Function
15:12	IC2F	RW	0	Input capture 2 filter
11:10	IC2PSC	RW	0	Input capture 2 prescaler
9:8	CC2S	RW	0	Capture/Compare 2 selection These 2 bits define the direction of the channel (input/output) and the selection of the input pins: 00: The CC2 channel is configured as an output; 01: CC2 channel is configured as input, IC2 is mapped on TI2; 10: CC2 channel is configured as input, IC2 is mapped on TI1;

Bit	Name	R/W	Reset Value	Function
				11: CC2 channel is configured as an input and IC2 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).
7:4	IC1F	RW	0	Input capture 1 filter These bits define the sampling frequency of the TI1 input and the length of the digital filter. The digital filter consists of an event counter that records N events and then generates a jump in the output: 0000: no filter, sampling at fDTS 1000: sampling frequency fSAMPLING=fDTS/8, N=6 0001: Sampling frequency fSAMPLING=fCK_INT, N=2 1001: Sampling frequency fSAMPLING=fDTS/8, N=8 0010: Sampling frequency fSAMPLING=fCK_INT, N=4 1010: Sampling frequency fSAMPLING=fDTS/16, N=5 0011: Sampling frequency fSAMPLING=fCK_INT, N=8 1011: Sampling frequency fSAMPLING=fDTS/16, N=6 0100: Sampling frequency fSAMPLING=fDTS/2, N=6 1100: Sampling frequency fSAMPLING=fDTS/16, N=8 0101: Sampling frequency fSAMPLING=fDTS/2, N=8 1101: Sampling frequency fSAMPLING=fDTS/32, N=5 0110: Sampling frequency fSAMPLING=fDTS/4, N=6 1110: Sampling frequency fSAMPLING=fDTS/32, N=6 0111: Sampling frequency fSAMPLING=fDTS/4, N=8 1111: Sampling frequency fSAMPLING=fDTS/32, N=8
3:2	IC1PSC	RW	0	Input capture 1 prescaler These 2 bits define the prescaler factor for the CC1 input (IC1). Once CC1E=0 (in the TIMx_CCER register), the prescaler is reset. 00: No prescaler, capture is triggered once for every edge detected on the capture input port; 01: one capture triggered for every 2 events; 10: a capture is triggered every 4 events; 11: a capture is triggered every 8 events.
1:0	CC1S	RW	0	Capture/Compare 1 Selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: The CC1 channel is configured as an output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: CC1 channel is configured as input, IC1 is mapped on TI2; 11: CC1 channel is configured as input and IC1 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC1S is writable only when the channel is off (CC1E=0 in the TIMx_CCER register).

14.3.8. TIM2/3/4/5 Capture/Compare Mode Register 2 (TIMx_CCMR2)

Address offset:0x1C

Reset value:0x0000

Refer to the description of the CCMR1 register above

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

14.3.8.1. Output comparison mode

Bit	Name	R/W	Reset Value	Function
15	OC4CE	RW	0	Output Compare 4 clear enable
14:12	OC4M	RW	0	Output Compare 4 mode
11	OC4PE	RW	0	Output Compare 4 preload enable
10	OC4FE	RW	0	Output Compare 4 fast enable
9:8	CC4S	RW	0	Capture/Compare 4 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC4 channel is configured as output; 01: CC4 channel is configured as input, IC4 is mapped on TI4; 10: CC4 channel is configured as input, IC4 is mapped on TI3; 11: CC4 channel is configured as an input and IC4 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC4S is writable only when the channel is off (CC4E=0 in the TIMx_CCER register).
7	OC3CE	RW	0	Output Compare 1clear enable
6:4	OC3M	RW	0	Output Compare 3 mode
3	OC3PE	RW	0	Output Compare 3 preload enable
2	OC3FE	RW	0	Output Compare 3 fast enable
1:0	CC3S	RW	0	Capture/Compare 3 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC3 channel is configured as output; 01: CC3 channel is configured as input, IC3 is mapped on TI3; 10: CC3 channel is configured as input, IC3 is mapped on TI4;

Bit	Name	R/W	Reset Value	Function
				11: CC3 channel is configured as an input and IC3 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC3S is writable only when the channel is off (CC3E=0 in the TIMx_CCER register).

14.3.8.2. Input capture mode

Bit	Name	R/W	Reset Value	Function
15:12	IC4F	RW	0	Input capture 4 filter
11:10	IC4PSC	RW	0	Input capture 4 prescaler
9:8	CC4S	RW	0	Capture/Compare 4 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC4 channel is configured as an output; 01: CC4 channel is configured as input, IC4 is mapped on TI4; 10: CC4 channel is configured as input, IC4 is mapped on TI3; 11: CC4 channel is configured as input and IC4 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC4S is writable only when the channel is off (CC4E=0 in the TIMx_CCER register).
7:4	IC3F	RW	0	Input capture 3 filter
3:2	IC3PSC	RW	0	Input capture 3 prescaler
1:0	CC3S	RW	0	Capture/Compare 3 Selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC3 channel is configured as output; 01: CC3 channel is configured as input, IC3 is mapped on TI3; 10: CC3 channel is configured as input, IC3 is mapped on TI4; 11: CC3 channel is configured as an input and IC3 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC3S is writable only when the channel is off (CC3E=0 in the TIMx_CCER register).

14.3.9. TIM2/3/4/5 Capture/Compare Enable Registers (TIMx_CCER)

Address offset:0x20

Reset value:0x0000

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

Reserved	CC4P	CC4E	Reserved	CC3P	CC3E	Reserved	CC2P	CC2E	Reserved	CC1P	CC1E
Reserved	RW	RW	Reserved	RW	RW	Reserved	RW	RW	Reserved	RW	RW

Bit	Name	R/W	Reset Value	Function
15:14	Reserved, always 0			
13	CC4P	RW	0	Capture/Compare 4 output polarity Refer to the description of CC1P.
12	CC4E	RW	0	Capture/Compare 4 output enable Refer to the description of CC1E.
11:10	Reserved			
9	CC3P	RW	0	Capture/Compare 3 output polarity Refer to the description of CC1P.
8	CC3E	RW	0	Capture/Compare 3 output enable Refer to the description of CC1E.
7:6	Reserved			
5	CC2P	RW	0	Capture/Compare 2 output polarity Refer to the description of CC1P.
4	CC2E	RW	0	Capture/Compare 2 output enable Refer to the description of CC1E.
3:2	Reserved			
1	CC1P	RW	0	Capture/Compare 1 output polarity The CC1 channel is configured as an output: 0: OC1 active high; 1: OC1 active low. The CC1 channel is configured as an input: This bit selects whether IC1 or the inverted signal of IC1 is used as the trigger or capture signal. 0: not inverted: capture occurs on the rising edge of IC1; when used as an external trigger, IC1 is not inverted. 1: Inverted: capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted.
0	CC1E	RW	0	Input/Compare 1 output enable CC1 channel is configured as an output: 0: Disable - OC1 disables output. 1: Enable - OC1 signal output to the corresponding output pin. The CC1 channel is configured as an input: This bit determines whether the counter value can be captured into the TIMx_CCR1 register. 0: Capture disable;

Bit	Name	R/W	Reset Value	Function
				1: capture enable.

Table 14-3 Output Controls for Standard OCx Channels

CCXEBit	OCx output status
0	Output disabled (OCx=0, OCx_EN=0)
1	OCx = OCxREF + polarity，OCx_EN=1

Note: The status of the external I/O pins connected to the standard OCx channel depends on the OCx channel status and the GPIO and AFIO registers.

14.3.10. TIM2/3/4/5 counters (TIMx_CNT)

Address offset:0x24

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	Counter value

14.3.11. TIM2/3/4/5 prescalers (TIMx_PSC)

Address offset:0x28

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	PSC	RW	0	Prescaler value The counter clock frequency (CK_CNT) is equal to fCK_PSC/(PSC[15:0]+1). PSC contains the value loaded into the current prescaler register each time an update event is generated; an update event consists of a counter being

Bit	Name	R/W	Reset Value	Function
				cleared '0' by the UG bit of TIM_EGR or cleared '0' by a slave controller operating in reset mode.

14.3.12. TIM2/3/4/5 Automatic Reload Registers (TIMx_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	Prescaler value The ARR contains the value that will be loaded into the actual auto-reload register. Refer to Updates and Actions on ARR for details. The counter does not operate when the auto-reload value is empty.

14.3.13. TIM2/3/4/5 Capture/comparison register 1 (TIMx_CCR1)

Address offset:0x34

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR1	RW	0	Capture/Compare Channel 1 value If the CC1 channel is configured as an output: CCR1 contains the value loaded into the current Capture/Compare 1 register (preloaded value). If the preload function is not selected in the TIMx_CMR1 register (OC1PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare 1 register only when an update event occurs.

Bit	Name	R/W	Reset Value	Function
				The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on the OC1 port. If the CC1 channel is configured as an input: CCR1 contains the counter value transferred by the last input capture 1 event (IC1).

14.3.14. TIM2/3/4/5 Capture/comparison register 2 (TIMx_CCR2)

Address offset:0x38

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR2	RW	0	Capture/Compare Channel 1 value If the CC2 channel is configured as an output: CCR2 contains the value loaded into the current Capture/Compare 2 register (preloaded value). If the preload function is not selected in the TIMx_CMR2 register (OC2PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare2 register only when an update event occurs. The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on the OC2 port. If the CC2 channel is configured as an input: CCR2 contains the counter value transferred by the last input capture 2 event (IC2).

14.3.15. TIM2/3/4/5 Capture/comparison registers3 (TIMx_CCR3)

Address offset:0x3C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR3	RW	0	Capture/Compare 3 value of channel 3 If the CC3 channel is configured as an output: CCR3 contains the value loaded into the current Capture/Compare 3 register (preloaded value). If the preload function is not selected in the TIMx_CMR3 register (OC3PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare3 register only when an update event occurs. The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on the OC3 port. If the CC3 channel is configured as an input: CCR3 contains the counter value transferred by the last input capture 3 event (IC3).

14.3.16. TIM2/3/4/5 Capture/comparison registers4 (TIMx_CCR4)

Address offset:0x40

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4															
RW/RO															

Bit	Name	R/W	Reset Value	Function
15:0	CCR4	RW	0	Capture/Compare Channel 4 value If the CC4 channel is configured as an output: CCR4 contains the value loaded into the current Capture/Compare 4 register (preloaded value). If the preload function is not selected in the TIMx_CMR4 register (OC4PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare4 register only when an update event occurs. The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on the OC4 port. If the CC4 channel is configured as an input: CCR4 contains the counter value transferred by the last input capture 4 event (IC4).

14.3.17. TIM2/3/4/5 DMA Control Registers (TIMx_DCR)

Address offset:0x48

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:13	Reserved, always 0			
12:8	DBL	RW	0	DMA burst length These bits define the length of the DMA transfer in continuous mode (the timer performs one continuous transfer when a read or write is made to the TIMx_DMAR register), i.e.: defines the number of transfers, which can be half-words (double bytes) or bytes: 00000: 1 transmission 00001: 2 transfers 00010: 3 transfers ..... 10001: 18 transmissions Example: Let's consider this transmission: DBL=7, DBA=TIM2_CR1 - If DBL=7 and DBA=TIM2_CR1 indicates the address of the data to be transmitted, then the address of the transmission is given by the following equation: (address of TIMx_CR1) + DBA + (DMA index), where DMA index = DBL where (address of TIMx_CR1) + DBA plus 7 gives the address where the data will be written or read, so that the transfer of data will take place in the 7 registers starting from the address (address of TIMx_CR1) + DBA. Depending on the setting of the DMA data length, the following may occur: - If the data is set to half words (16 bits), then the data is transferred to all 7 registers. - If the data is set to byte, the data is still transferred to all 7 registers: the first register contains the first MSB byte, the second register contains the first LSB byte, and so on. Therefore for timers, the user must specify the width of the data to be transferred by the DMA.
7:5	Reserved, always reads 0.			
4:0	DBA	RW	0	DMA base address These bits define the base address of the DMA in continuous mode (when reading or writing to the TIMx_DMAR register), and the DBA is defined as the offset from the address where the TIMx_CR1 register is located: 00000: TIMx_CR1. 00001: TIMx_CR2.

Bit	Name	R/W	Reset Value	Function
				00010: TIMx_SMCR. ……

14.3.18. DMA address for TIM2/3/4/5 continuous mode (TIMx_DMAR)

Address offset:0x4C

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	DMAB	RW	0	DMA register for burst accesses A read or write to the TIMx_DMAR register results in an access operation to the register where the following address is located: TIMx_CR1 address + DBA + DMA index, where: "TIMx_CR1 address" is the address where control register 1 (TIMx_CR1) is located; "DBA" is the base address defined in the TIMx_DCR register; "DMA Index" is the offset automatically controlled by the DMA, which depends on the DBL defined in the TIMx_DCR register.

Note: When using the DMA continuous transfer function, the value of the CNDTR register of the corresponding channel in the DMA must correspond to the value of DBL in the TIMx_DCR register, otherwise this will not work properly.

14.3.19. TIM2/3/4/5 register mapping

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
	Read/W																							rw		rw	rw		rw	rw	rw	rw	rw

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	write																																
	Reset Value																							0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Reserved																								TI1S	MMS			CCDS	Reserved		
	Read/Write																									rw	rw			rw			
	Reset Value																									0	0			0			
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETPS		ETF				MSM	TS			Reserved	SMS		
	Read/Write																	rw	rw	rw		rw				rw	rw				rw		
	Reset Value																	0	0	0		0				0	0				0		
0x0C	TIMx_DIER	Reserved																TDE	Reserved	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE	
	Read/W																	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	write																																
	Reset Value																		0		0	0	0	0	0		0		0	0	0	0	0
0x10	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Read/Write																				rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0
	Reset Value																				0	0	0	0			0		0	0	0	0	0
0x14	TIMx_EGR	Reserved																									TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG
	Read/Write																										w		w	w	w	w	w
	Reset Value																										0		0	0	0	0	0
0x18	TIMx_CCMR1	Reserved																	OC2CE	OC2M		OC2PE	OC2FE	CC2S			OC1CE	OC1M		OC1PE	OC1FE	CC1S	
	Read/W																		IC2F			IC2PSC		rw			rw			IC1PSC		rw	
																			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	write	Reserved																															
	Reset Value																	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0x1C	TIMx_CCMR2	Reserved																OC4CE	OC4M			OC4PE	OC4FE	CC4S		OC3CE	OC3M			OC3PE	OC3FE	CC3S	
																		IC4F				IC4PSC				IC3F				IC3PSC			
	Read/Write																	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	
	Reset Value																	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0x20	TIMx_CCER	Reserved																		CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
	Read/Write																			rw	rw			rw	rw			rw	rw			rw	rw
	Reset Value																			0	0			0	0			0	0			0	0
0x24	TIMx_CNT	Reserved																CNT															
	Read																	rw															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	/Write																																
	Reset Value																	0															
0x28	TIMx_PSC																	PSC															
	Read/Write																	rw															
	Reset Value	Reserved																0															
0x2C	TIMx_ARR																	ARR															
	Read/Write																	rw															
	Reset Value	Reserved																0xFFFF															
0x34	TIMx_CCR1																	CCR1															
	Read/W	Reserved																rw/ro															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	write																																
	Reset Value																	0															
0x38	TIMx_CCR2																	CCR2															
	Read/Write																	rw/ro															
	Reset Value																	0															
0x3C	TIMx_CCR3																	CCR3															
	Read/Write																	rw/ro															
	Reset Value																	0															
0x40	TIMx_CCR4																	CCR4															
	Read/W																	rw/ro															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	write	Reserved																															
	Reset Value																	0															
0x48	TIMx_DCR	Reserved																			DBL				Reserved				DBA				
	Read/Write																				rw								rw				
	Reset Value																				0								0				
0x4C	TIMx_DMAR	Reserved																DMAB															
	Read/Write																	rw															
	Reset Value																	0															

15. General-purpose timer (TIM9 and TIM12)

15.1. Introduction

The general purpose timer (TIM9 and TIM12) consists of a 16-bit auto-load counter driven by a programmable prescaler.

It is suitable for a variety of uses, including measuring the pulse width of an input signal (input capture), or generating an output waveform (output comparison, PWM).

Using the timer prescaler and the RCC clock to control the prescaler, pulse width and waveform period can be adjusted from a few microseconds to a few milliseconds.

The general-purpose timers TIM9 and TIM12 are completely independent; they do not share any resources and can operate synchronously.

15.1.1. TIM9 and TIM12 main features

The functions of the TIM9 and TIM12 timers include:

- 16-bit upward automatic counter loading
- 16-bit programmable (can be modified in real time) prescaler with any value between 1 and 65536 for the counter clock frequency dividing factor
- Up to 2 independent channels:
  - Input Capture
  - Output comparison
  - PWM generation (edge-aligned mode)
  - Single pulse mode output
- Synchronization circuit to control the interconnection between timers by external signals
- Interrupt are generated when the following events occur:
  - Update: counter up overflow, counter initialization (triggered by software or internally)
  - Trigger events (counter start, stop, initialize or count triggered internally)
  - Input capture
  - Output comparison

15.1.2. Module Block Diagram

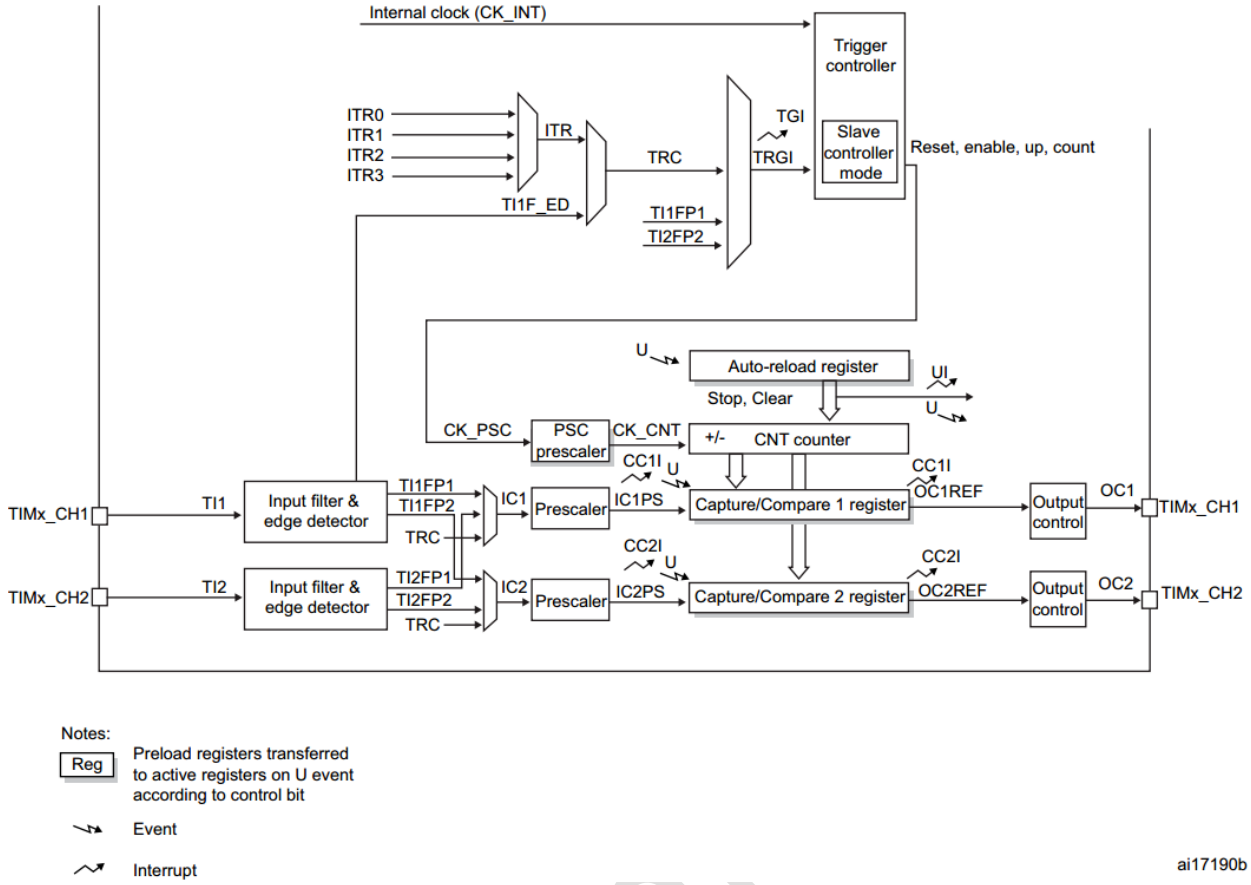


Figure 15-1 TIM9 and TIM12 modules

15.2. TIM9 and TIM12 Functional Descriptions

15.2.1. Time base unit

The main part of the general purpose timer is a 16-bit counter and its associated auto-load register. This counter can count upwards.

The clock of the counter can be divided by a prescaler.

The counter, the autoloader register and the prescaler register can be read and written by software, even if the counter is still running.

The time base unit contains:

- Counter register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto Load Register (TIMx_ARR)

The autoload register is preloaded, and writing or reading the autoreload register will access its preload register. Depending on the setting of the Auto Reload Preload Enable bit (ARPE) in the TIMx_CR1 register, the contents of the preload register are transferred to the shadow register immediately or at each update event (UEV). An update event is generated when the counter reaches the overflow condition and when the UDIS bit in the TIMx_CR1 register equals 0. Update events can also be generated by software pieces. The generation of update events in each configuration will be described in detail later.

The counter is driven by the prescaler divided clock output CK_CNT, which is valid for the counter only when the counter enable bit (CEN) in the TIMx_CR1 register is set. (See the description of the slave mode controller for more details on enabling the counter).

Note, that the counter starts counting one clock cycle after the CEN bit of the TIMx_CR register is set.

15.2.1.1. Prescaler description

The prescaler can divide the counter clock frequency by any value between 1 and 65536. It is based on a 16-bit counter controlled by a 16-bit register (in the TIMx_PSC register). Because this control register has a buffer, it is able to be changed at runtime. The new prescaler parameters will be adopted when the next update event comes.

An example of changing the counter parameters while the prescaler is running is shown below.

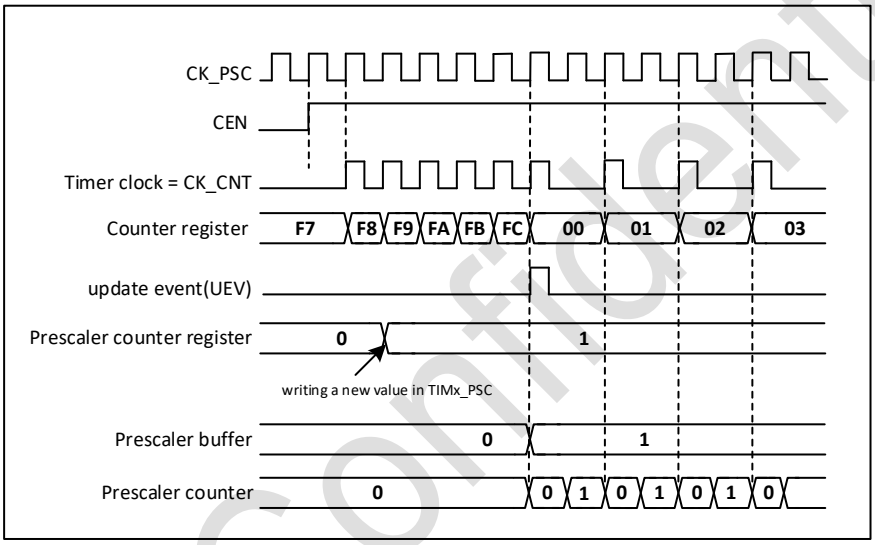


Figure 15-2 Counter timing diagram with prescaler division change from 1 to 2

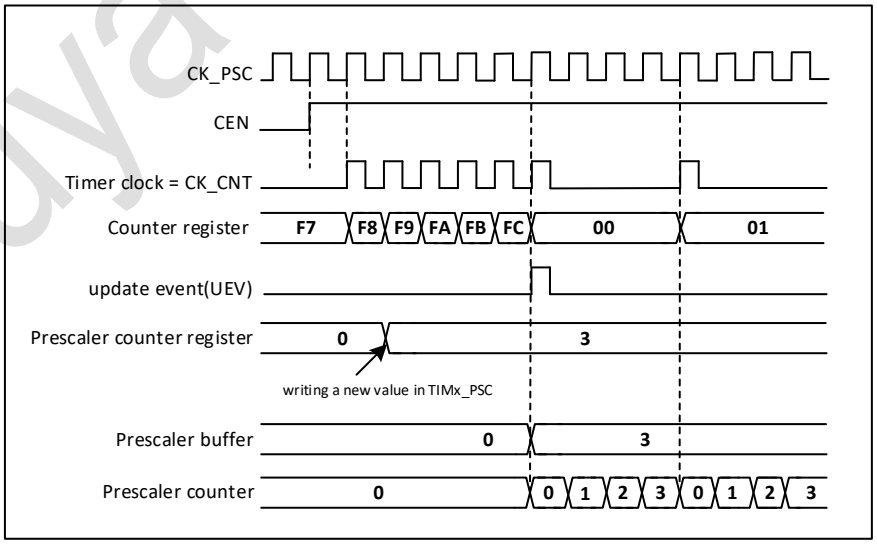


Figure 15-3 Counter timing diagram with prescaler division change from 1 to 4

15.2.2. Counter modes

15.2.2.1. Upcounting mode

In up-count mode, the counter counts from 0 to the autoload value (the contents of TIMx_ARR), then starts counting from 0 again and generates a count overflow event.

An update event is generated for each count overflow.

Setting the UG bit in the TIMx_EGR register (either in software or using the slave mode controller) also generates an update event.

The update event can be disabled by setting the UDIS bit in the TIMx_CR1 register; this prevents the shadow register from being updated when a new value is written to the preload register. An update event will not be generated until the UDIS bit is cleared '0'. Even then, the counter will still be cleared '0' when an update event should be generated, and the internal counter of the prescaler is also cleared '0' (but the value of the prescaler remains unchanged).

In addition, if the URS bit (Select Update Request Source) in the TIMx_CR1 register is set, an update event UEV can be generated by setting the UG bit, but the UIF flag bit will not be set up (i.e., no interrupt request will be generated). This is to avoid clearing the counter on a capture event and generating both an update and a capture interrupt.

When an update event occurs, all of the following registers are updated and the hardware sets the update flag bit (UIF bit in the TIMx_SR register) simultaneously (based on the URS bit):

- The autoload shadow register is reset to the value of the preload register (TIMx_ARR).
- The prescaler buffer is reset to the value of the preload register (the contents of the TIMx_PSC register).

The following figure gives some examples of how the counter acts at different clock frequencies when TIMx_ARR=0x36.

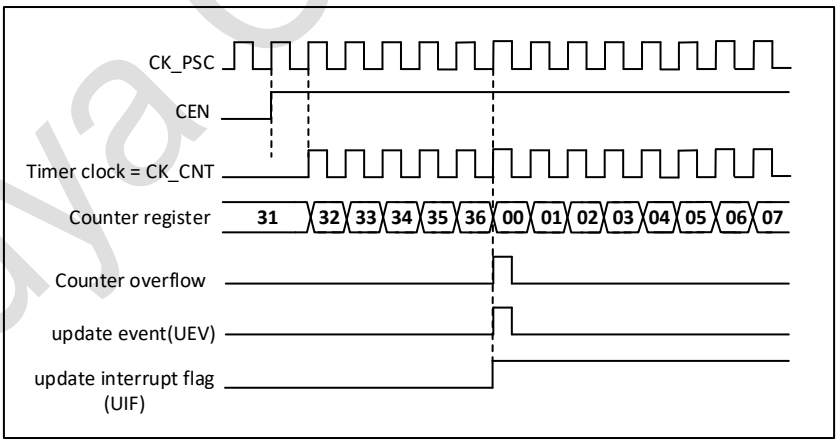


Figure 15-4 Counter timing diagram, internal clock divided by 1

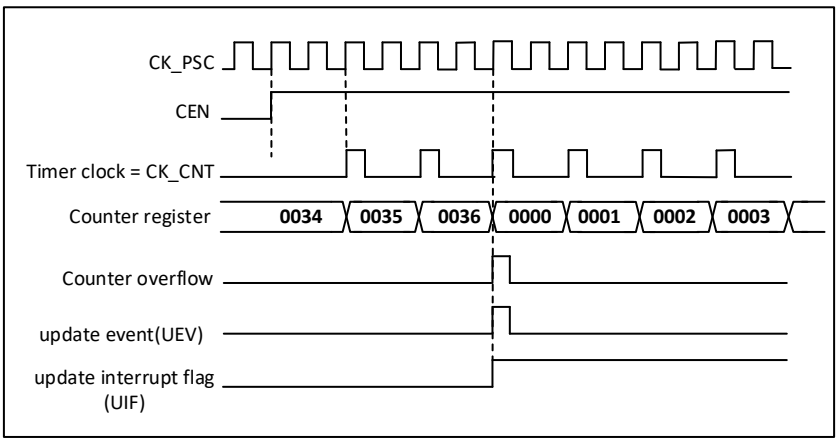


Figure 15-5 Counter timing diagram, internal clock divided by 2

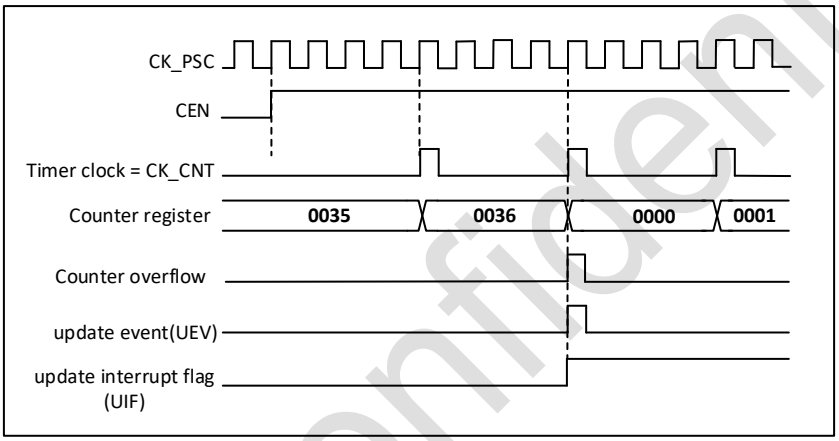


Figure 15-6 Counter timing diagram, internal clock divided by 4

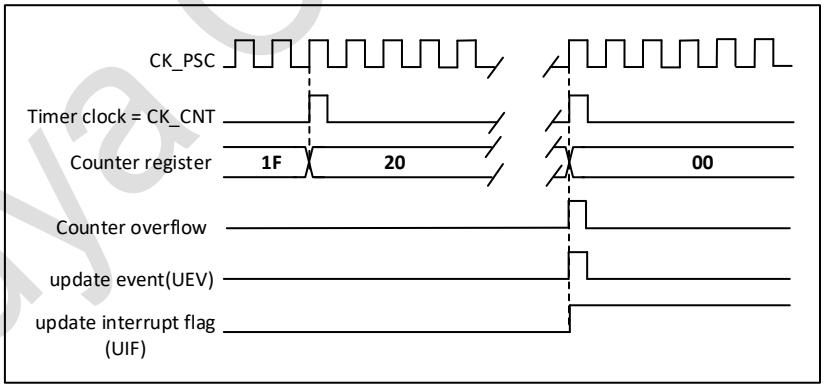


Figure 15-7 Counter timing diagram, internal clock divided by N

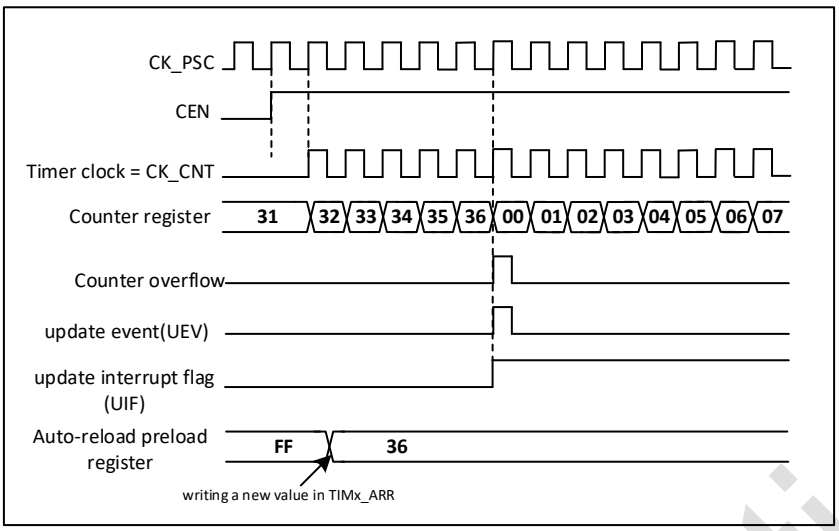


Figure 15-8 Counter timing diagram, Update event when ARPE = 0(TIM3_ARR not preloaded)

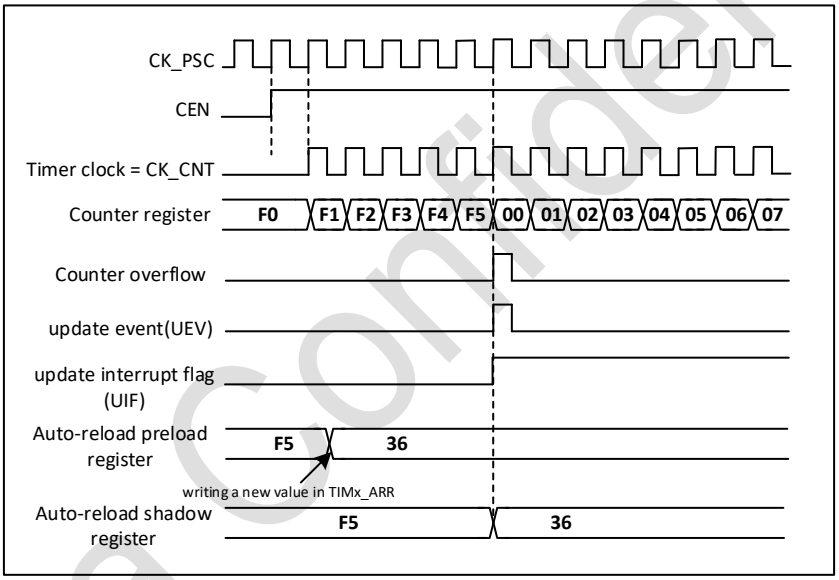


Figure 15-9 Counter timing diagram, Update event when ARPE = 1(TIM3_ARR preloaded)

15.2.3. Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode 1: external input pin (TIx)
- Internal trigger input (ITRx): Use one timer as a prescaler for another timer. For example, one timer Timer1 can be configured while acting as a prescaler for another timer Timer2.

15.2.3.1. Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), the CEN and UG bits (TIMx_EGR register) are the de facto control bits and can only be modified by software (the UG bit is still automatically cleared). As long as the CEN bit is written to '1', the prescaler clock is provided by the internal clock CK_INT.

The following diagram shows the operation of the control circuit and the up counter in general mode without the prescaler.

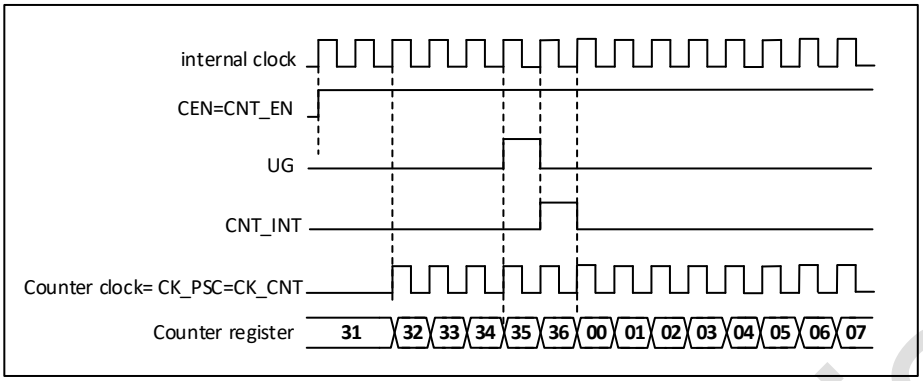


Figure 15-10 Control circuit in general mode with internal clock division factor of 1

15.2.3.2. External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count on each rising or falling edge of the selected input.

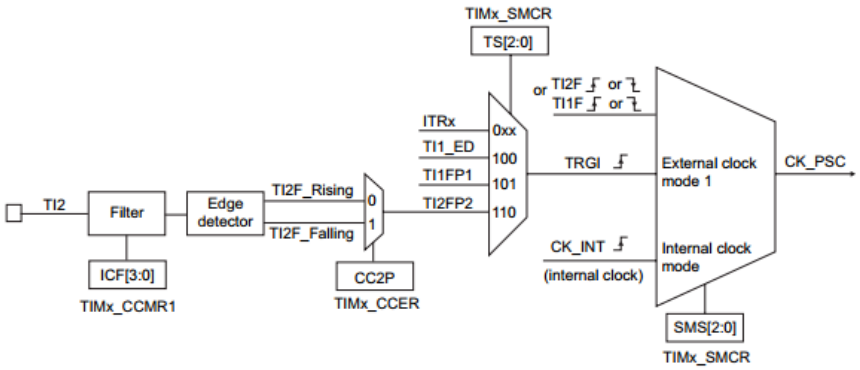


Figure 15-11 TI2 external clock connection example

For example, to configure the counter to count upward on the rising edge of the T12 input, use the following steps:

1. Configure the TIMx_CCMR1 register CC2S=01 so that channel 2 detects the rising edge of the TI2 input;
2. Configure the TIMx_CCMR1 register IC2F[3:0] to select the input filter bandwidth (if no filter is required, keep IC2F=0000);
3. Configure the TIMx_CCER register with CC2P=0 to select the rising edge polarity;
4. Configure SMS=111 in the TIMx_SMCR register to select the timer as external clock mode 1;
5. Configure TS=110 in the TIMx_SMCR register to select TI2 as the trigger input source;
6. Set CEN=1 in TIMx_CR1 register to start the counter.

Note: The capture prescaler is not used as a trigger, so there is no need to configure it.

When the rising edge occurs at TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge at TI2 and the actual clock of the counter depends on the resynchronization circuit at the TI2 input.

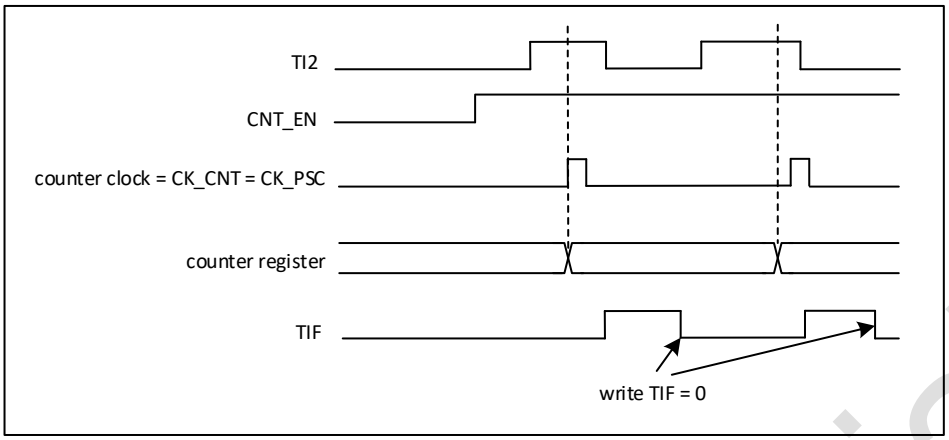


Figure 15-12 Control circuit in external clock mode 1

15.2.4. Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

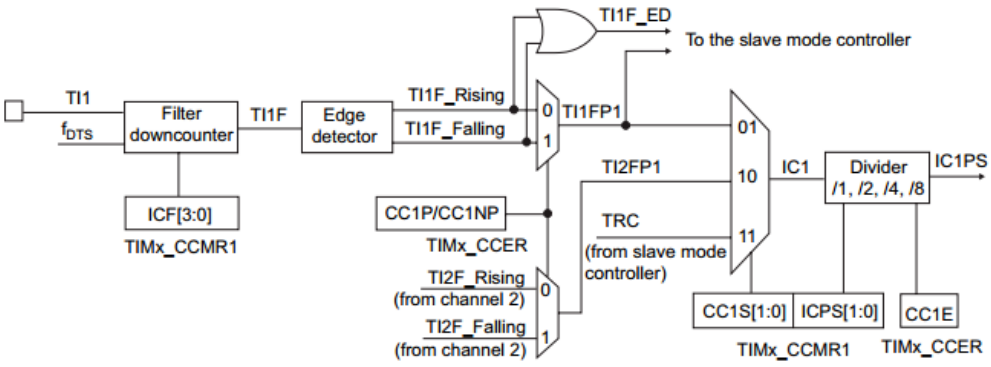


Figure 15-13 Capture/compare channel (example: channel 1 input stage)

The output stage generates an intermediate waveform which is then used for reference of OCxRef (active high). The polarity acts at the end of the chain.

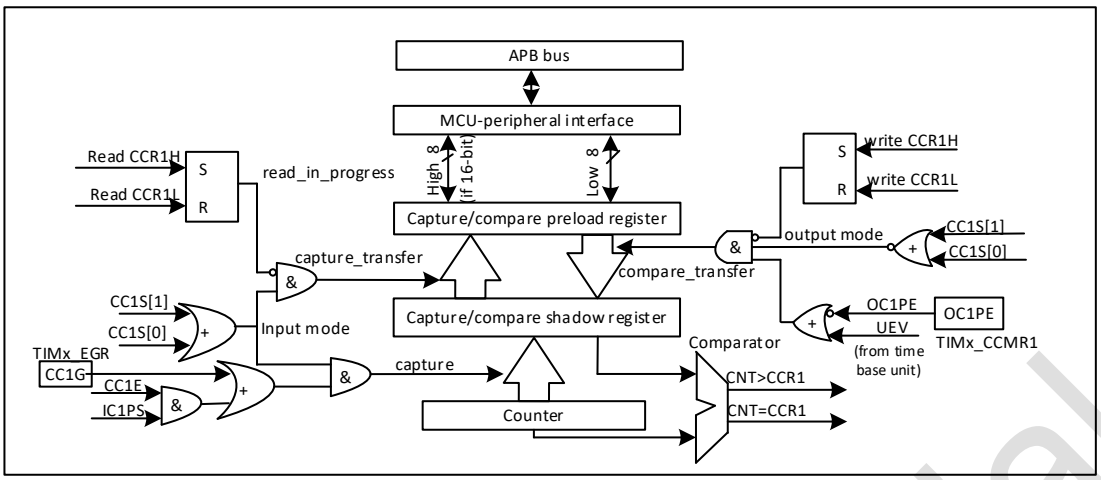


Figure 15-14 Capture/compare channel 1 main circuit

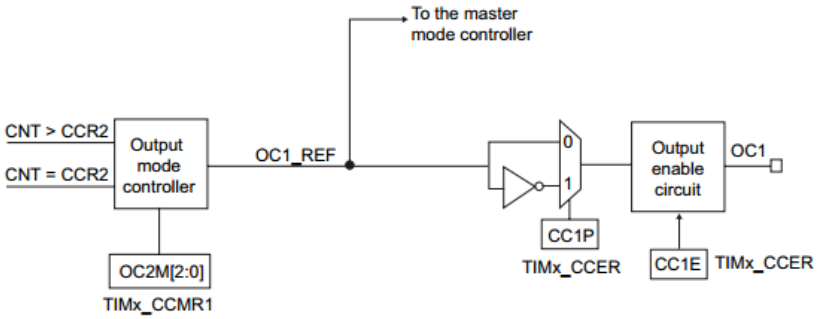


Figure 15-15 Output stage of capture/compare channel (channel 1)

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

15.2.5. Input capture mode

In the input capture mode, the current value of the counter is latched into the capture/compare register (TIMx_CCRx) when the corresponding edge on the ICx signal is detected. When a capture event occurs, the corresponding CCxIF flag (TIMx_SR register) is set to 1. If an interrupt operation is open, an interrupt request will be generated. If the CCxIF flag is already high when a capture event occurs, the over-capture flag CCxOF (TIMx_SR register) is set to 1. CCxIF can be cleared by writing CCxIF=0, or by reading the capture data stored in the TIMx_CCRx register. CCxOF can be cleared by writing CCxOF=0.

The following example shows how to capture the counter value into the TIMx_CCR1 register on the rising edge of the TI1 input, as follows:

- Selecting a valid input: TIMx_CCR1 must be connected to the TI1 input, so write CC1S=01 in the TIMx_CCMR1 register, as long as CC1S is not '00', the channel is configured as an input and the TIMx_CCR1 register becomes read-only.
- The input filter is configured for the desired bandwidth according to the characteristics of the input signal (i.e., when the input is TIx, the input filter control bit is the ICxF bit in the TIMx_CCMRx register). Assuming that the input signal dithers for a maximum of 5 internal clock cycles, we shall configure the filter with a bandwidth longer than 5 clock cycles; thus we can sample 8 times in a row (at fDTS frequency) to confirm one true edge transition on TI1, i.e., write IC1F=0011 in the TIMx_CCMR1 register.
- Select the valid transition edge of the TI1 channel by writing 00 (set to rising edge) to the CC1P and CC1NP bits in the TIMx_CCER register.
- Configure the input prescaler. In this example, we want the capture to occur at every valid level transition moment, so the prescaler is disabled (write IC1PS=00 in the TIMx_CCMR1 register).
- Setting the TIMx_CCER register with CC1E=1 allows the capture counter value to the capture register.
- If required, the associated interrupt request can be allowed by setting the CC1IE bit in the TIMx_DIER register.

When a capture of an input occurs:

- When a valid level transition is generated, the value of the counter is transferred to the TIMx_CCR1 register.
- The CC1IF flag bit is set (interrupt flag). When at least 2 consecutive captures occur and CC1IF has not been cleared, CC1OF is also set to 1.
- If the CC1IE bit is set, an interrupt will be generated.

In order to handle overcatch, it is recommended to read the data before the overcatch flag is raised. This is to avoid losing capture overflow information that may be generated after the capture overflow flag is read and before the data is read.

Note: Setting the corresponding CCxG bit in the TIMx_EGR register allows you to generate an input capture interrupt request by software.

15.2.6. PWM input mode

This mode is a special case of input capture mode and operates the same as input capture mode except for the following differences:

- The two ICx signals are mapped to the same TIx input.
- The 2 ICx signals are edge active, but of opposite polarity.
- One of the TIxFP signals is used as the trigger input signal, while the slave mode controller is configured to reset mode.

For example, the user can measure the period (TIMx_CCR1 register) and duty cycle (TIMx_CCR2 register) of the PWM signal input to TI1, as follows (depending on the frequency of CK_INT and the value of the prescaler).

- Select valid input for TIMx_CCR1: Set CC1S=01 of TIMx_CCMR1 register (TI1 selected).
- Select valid polarity of TI1FP1 (used to capture data into TIMx_CCR1 and clear the counter): Set CC1P and CC1NP to 00 (rising edge valid).
- Select valid input for TIMx_CCR2: Set CC2S=10 of TIMx_CCMR1 register (TI1 selected).
- Select valid polarity of TI1FP2 (capture data to TIMx_CCR2): Set CC2P and CC2NP to 10 (falling edge valid).
- Select a valid trigger input signal: Set TS=101 in TIMx_SMCR register (select TI1FP1).
- Configure the slave mode controller to reset mode: Set SMS=100 in TIMx_SMCR.
- Enable capture: Set CC1E=1 and CC2E=1 in the TIMx_CCER register.

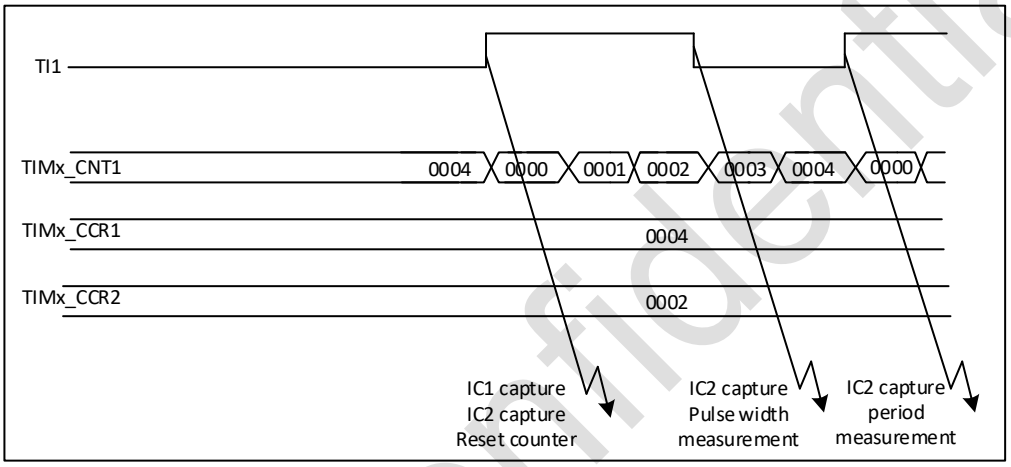


Figure 15-16 PWM Input Mode Timing

Since only TI1FP1 and TI2FP2 are connected to the slave mode controller, only the TIMx_CH1/TIMx_CH2 signals can be used for the PWM input mode.

15.2.7. Force output mode

In the output mode (CCxS=00 in the TIMx_CCMRx register), the output compare signal (OCxREF and the corresponding OCx/OCxN) can be directly forced by software to the valid or invalid state, regardless of the comparison result between the output compare register and the counter.

The output comparison signal (OCxREF/OCx) can be forced to be valid by setting the corresponding OCxM=101 in the TIMx_CCMRx register. In this way, OCxREF is set high (OCxREF is always active high), and OCx gets a signal of opposite polarity of CCxP.

For example: CCxP=0 (OCx is active high), then OCx is set strongly high.

Setting OCxM=100 in the TIMx_CCMRx register can force the OCxREF signal low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter is still going on, and the corresponding flags are modified. Therefore the corresponding interrupt request will still be generated. This will be described in the Output Compare Mode section below.

15.2.8. Output compare mode

This function is used to control an output waveform, or to indicate that a given period of time has expired.

When the counter is the same as the contents of the capture/compare register, the output compare function does the following:

- The values defined by the output compare mode (OCxM bit in the TIMx_CCMRx register) and output polarity (CCxP bit in the TIMx_CCER register) are output to the corresponding pins. When comparing matches, the output pin can hold its level (OCxM=000), be set to a valid level (OCxM=001), be set to an invalid level (OCxM=010), or be flipped (OCxM=011).
- Set the flag bit in the interrupt status register (CCxIF bit in the TIMx_SR register).
- If the corresponding interrupt mask (CCxIE bit in TIMx_DIER register) is set, an interrupt is generated.

You can select whether the TIMx_CCRx register needs to use a preload register by configuring the OCxPE bit in TIMx_CCMRx.

In output compare mode, the update event UEV has no effect on the OCxREF and OCx outputs. The synchronization can be done with an accuracy of one count cycle of the counter. The output compare mode (in single pulse mode) can also be used to output a single pulse.

Configuration steps for output comparison mode:

1. Select the counter clock (internal, external, prescaler).
2. Write the corresponding data into the TIMx_ARR and TIMx_CCRx registers.
3. If an interrupt request is to be generated, set the CCxIE bit.
4. Select the output mode, e.g:
   - Flip the output pins of OCx when the counter is required to match CCRx, set OCxM = 011
   - Set OCxPE = 0 to disable preload register
   - Set CCxP = 0 to select polarity to active high
   - Set CCxE = 1 to enable output
5. Set the CEN bit of the TIMx_CR1 register to start the counter

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not used (OCxPE='0', otherwise the shadow register of TIMx_CCRx can only be updated when the next update event occurs). An example is given in the following figure.

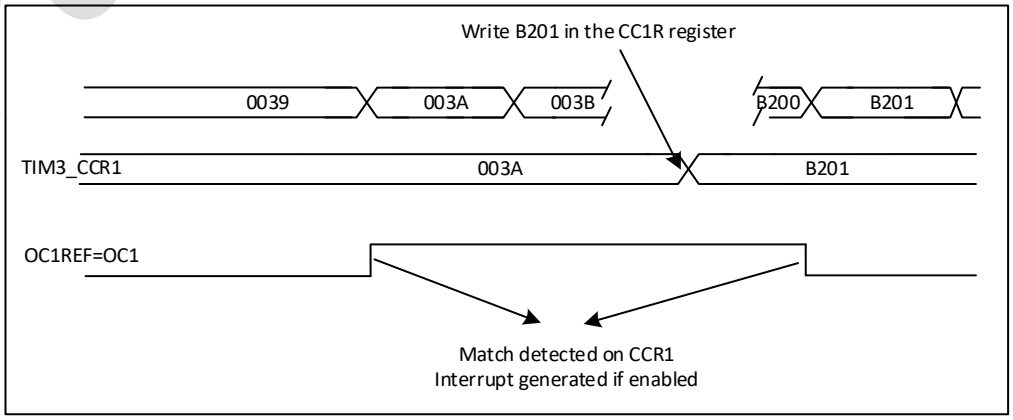


Figure 15-17 Output compare mode, toggle on OC1

15.2.9. PWM mode

The pulse width modulation mode can generate a signal with a frequency determined by the TIMx_ARR register and a duty cycle determined by the TIMx_CCRx register.

The OCxM bit in the TIMx_CCMRx register is written to '110' (PWM mode 1) or '111' (PWM mode 2) to be able to independently set each OCx output channel to generate one PWM. it must be enabled by setting the The corresponding preload registers must be enabled by setting the OCxPE bit of the TIMx_CCMRx register and finally by setting the ARPE bit of the TIMx_CR1 register to enable the preload registers for automatic reloading (in upcount or centrosymmetric mode).

The preload registers can be transferred to the shadow registers only when an update event occurs, so the user must initialize all registers by setting the UG bit in the TIMx_EGR register before the counter starts counting.

The polarity of OCx can be set by software through the CCxP bit in the TIMx_CCER register, which can be set to active high or active low. the output enable of OCx is controlled through the CCxE bit (in TIMx_CCER). See the description of TIMx_CCER register for details.

In PWM mode (mode 1 or mode 2), TIMx_CNT and TIMx_CCRx are always being compared, (based on the counter count direction) to determine if TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx is met.

The timer is capable of generating edge-aligned PWM signals.

15.2.9.1. PWM edge-aligned mode

- Upcounting configuration

The following is an example of PWM mode 1. When TIMx_CNT<TIMx_CCRx, PWM reference signal OCxREF is high, otherwise it is low. If the comparison value in TIMx_CCRx is greater than the auto reload value (TIMx_ARR), OCxREF remains '1'. If the comparison value is 0, OCxREF is kept as '0'. The following figure shows an example of edge-aligned PWM waveform when TIMx_ARR=8.

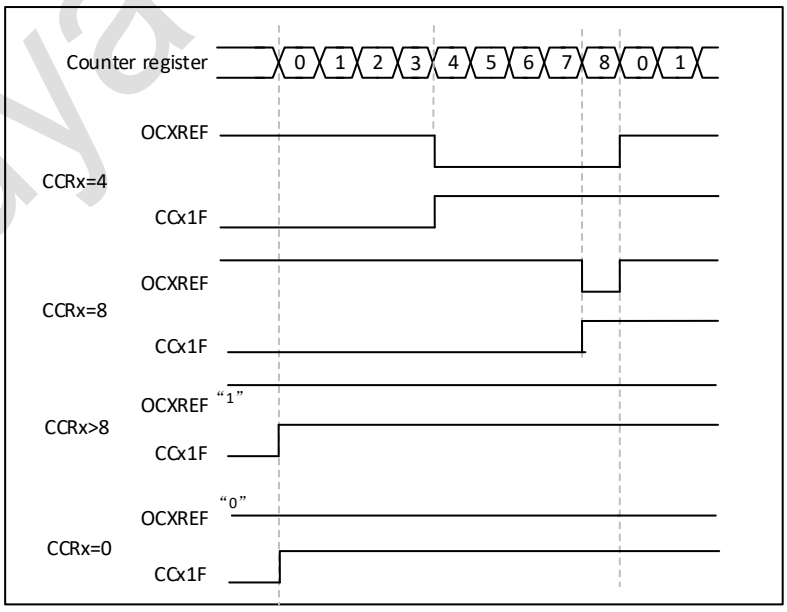


Figure 15-18 Edge-aligned PWM waveforms (ARR = 8)

15.2.10. One-pulse mode

The single pulse mode (OPM) is a special case of the many modes described earlier. This mode allows the counter to respond to an excitation and generate a pulse with a programmable pulse width after a programmable delay.

The counter can be started from the mode controller to generate waveforms in the output compare mode or PWM mode. Setting the OPM bit in the TIMx_CR1 register will select the single pulse mode, which allows the counter to automatically stop when the next update event UEV is generated.

A pulse can be generated only when the comparison value is different from the initial value of the counter. Before starting (when the timer is waiting to be triggered), it must be configured as follows:

- Upward counting mode: counter CNT < CCRx ≤ ARR (in particular, 0 < CCRx).

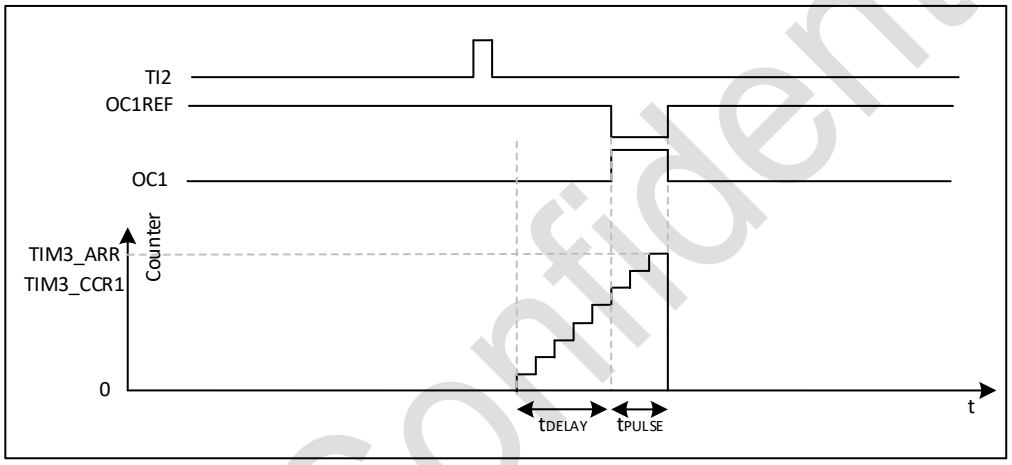


Figure 15-19 Example of one-pulse mode

For example, you need to generate a positive pulse of length tPULSE on OC1 after a delay of tDELAY starting from a rising edge detected on the TI2 input pin.

Assume TI2FP2 as the trigger.

- Set CC2S=01 in the TIMx_CCMR1 register to map TI2FP2 to TI2.
- Set CC2P and CC2NP=00 in TIMx_CCER register to enable TI2FP2 to detect rising edge.
- Set TS=110 in TIMx_SMCR register to trigger the TI2FP2 as a slave mode controller (TRGI).
- Set SMS=110 in TIMx_SMCR register (trigger mode), TI2FP2 is used to start the counter.

The waveform of the OPM is determined by the value written to the comparison register (taking into account the clock frequency and the counter prescaler)

- tDELAY is defined by the value in the TIMx_CCR1 register.
- tPULSE is defined by the difference between the autoload value and the compare value (TIMx_ARR - TIMx_CCR1).
- It is assumed to generate a waveform from 0 to 1 when a comparison match occurs and a waveform from 1 to 0 when the counter reaches the preload value; first set OC1M = 111 in the TIMx_CCMR1 register and enter PWM mode 2; selectively enable the preload registers as needed: set OC1PE = 1 in TIMx_CCMR1 and the TIMx_CR1 register Then fill in the

comparison value in TIMx_CCR1 register and the auto-load value in TIMx_ARR register, set the UG bit to generate an update event, and wait for an external trigger event on TI2. In this example, CC1P=0.

Since only one pulse is required, OPM=1 in the TIMx_CR1 register must be set to stop counting on the next update event (when the counter flips from the auto-load value to 0). When OPM=0, repeat mode is selected.

15.2.10.1. Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay tDELAY.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the com-parison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is con-figured in PWM1 or PWM2 mode.

15.2.11. Synchronization of TIMx timers and external triggers

The TIMx timer can be synchronized with an external trigger in several modes: reset mode, gated mode and trigger mode.

15.2.11.1. Slave mode: Reset mode

On the occurrence of a trigger input event, the counter and its prescaler can be reinitialized; at the same time, if the UDIS bit of the TIMx_CR1 register is low, an update event UEV is also generated; then all preload registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the rising edge of the TI1 input causes the up counter to be cleared to zero:

- Configure channel 1 to detect the rising edge of TI1. Configure the bandwidth of the input filter (in this example, no filter is needed, so keep IC1F=0000). No capture prescaler is used in the trigger operation, so no configuration is required. the CC1S bit selects only the input capture source, i.e., CC1S=01 in the TIMx_CCMR1 register. set CC1P and CC1NP to 00 in the TIMx_CCER register to determine the polarity (rising edge detection only).
- Set SMS=100 in TIMx_SMCR register to configure the timer to reset mode; set TS=101 in TIMx_SMCR register to select TI1 as input source.
- Set CEN=1 in TIMx_CR1 register to start the counter.

The counter starts counting based on the internal clock and then operates normally until a rising edge appears on TI1; at this point, the counter is cleared and then starts counting from 0 again. At the same time, the trigger flag (TIF bit in TIMx_SR register) is set and an interrupt request is generated according to the setting of the TIE (interrupt enable) bit, in the TIMx_DIER register.

The following figure shows the action when the Auto Reload register TIMx_ARR=0x36. The delay between the rising edge of TI1 and the actual reset of the counter depends on the resynchronization circuitry at the TI1 input.

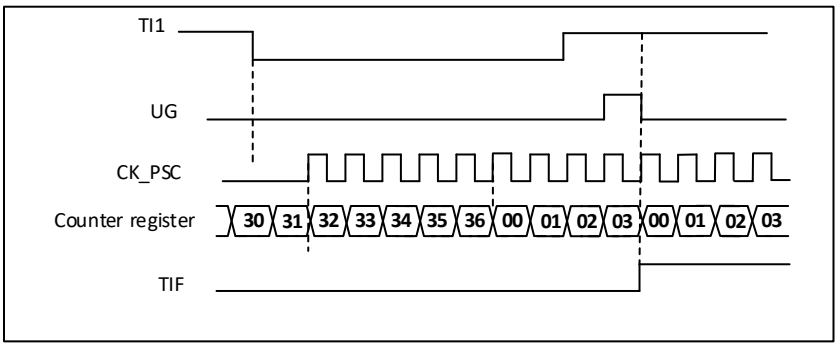


Figure 15-20 Control circuit in reset mode

15.2.11.2. Slave mode: Gated mode

Enables the counter according to the selected input level.

In the following example, the counter only counts up when TI1 is low:

- Configure channel 1 to detect a low level on TI1. Configure the input filter bandwidth (in this example, no filtering is required, so keep IC1F=0000). No capture prescaler is used in the trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source by setting CC1S=01 in the TIMx_CCMR1 register. set CC1P and CC1NP to 10 in the TIMx_CCER register to determine the polarity (detect low level only).
- Set SMS=101 in TIMx_SMCR register to configure the timer to gated mode; set TS=101 in TIMx_SMCR register to select TI1 as input source.
- Set CEN=1 in TIMx_CR1 register to start the counter. In the gated mode, if CEN=0, the counter cannot start, regardless of the trigger input level.

As long as TI1 is low, the counter starts counting according to the internal clock, and stops counting once TI1 becomes high. The TIF flag in TIMx_SR is set when the counter starts or stops.

The delay between the rising edge of TI1 and the actual stop of the counter depends on the resynchronization circuit at the TI1 input.

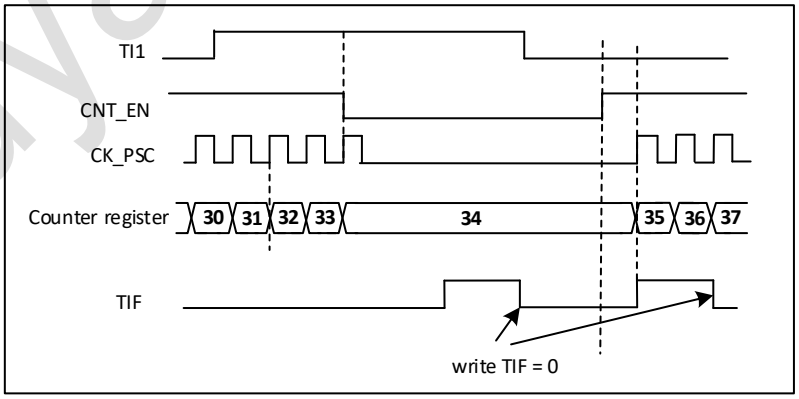


Figure 15-21 Control circuit in gated mode

15.2.11.3. Slave mode: Trigger mode

The event selected on the input enables the counter.

In the following example, the counter starts counting up at the rising edge of the TI2 input:

- Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is needed and IC2F = 0000 is held). No capture prescaler is used in the trigger operation, no configuration is required. the CC2S bit is only used to select the input capture source, set CC2S=01 in the TIMx_CCMR1 register. set CC2P and CC2NP to 10 in the TIMx_CCER register to determine the polarity (detect low level only).
- Set SMS=110 in TIMx_SMCR register to configure the timer to trigger mode; set TS=110 in TIMx_SMCR register to select TI2 as input source.

When there is a rising edge of TI2, the counter starts counting under the internal clock drive and the TIF flag is set at the same time.

The delay between the rising edge of TI2 and the counter starting to count depends on the resynchronization circuit at the TI2 input.

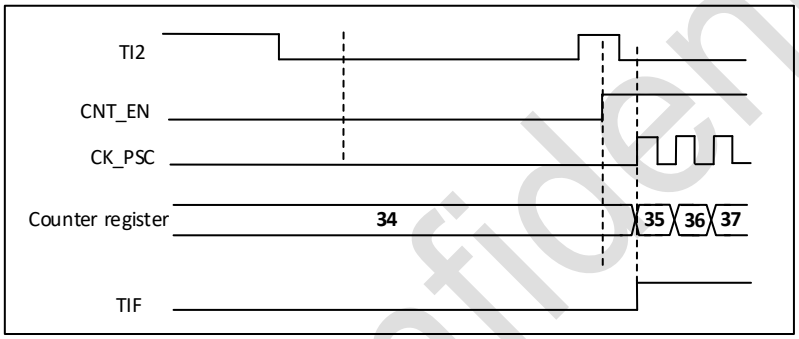


Figure 15-22 Control circuit in trigger mode

15.2.12. Timer synchronization

All TIMx timers are internally connected for timer synchronization or linking. When a timer is in master mode, it can reset, start, stop or provide a clock to the counter of another timer in slave mode.

TIM9 and TIM12 can only be used as slave timers.

The following figure shows an overview of the trigger selection and master mode selection modules.

15.2.12.1. Use one timer as a prescaler for another timer

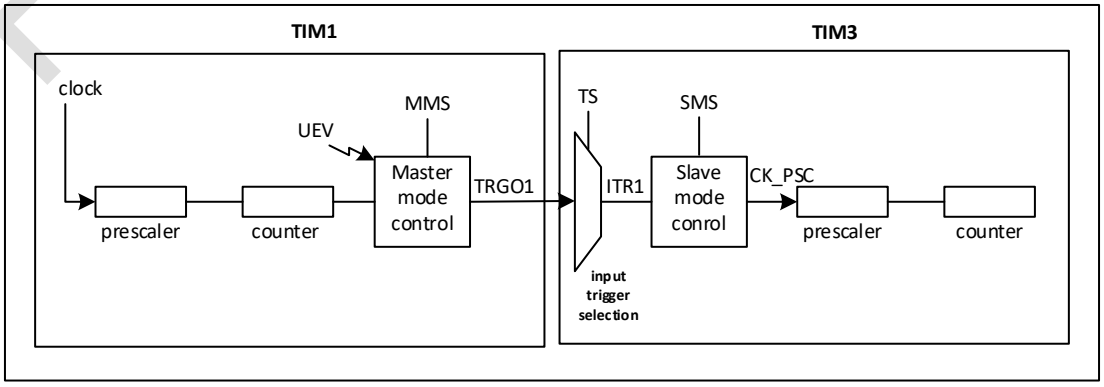


Figure 15-23 Master/Slave timer example

- Configure Timer 1 as the main mode, which can output a periodic trigger signal at every update event UEV. With MMS='010' of TIM1_CR2 register, output a rising edge signal on TRGO1 whenever an update event is generated.
- Connect the TRGO1 output of Timer 1 to Timer 2, set TS='000' of TIM2_SMCR register, and configure Timer 2 as a slave mode using ITR1 as the internal trigger.
- Then put the slave mode controller in external clock mode 1 (SMS=111 of TIM2_SMCR register); so that Timer 2 can be driven by the periodic rising edge (i.e. counter overflow of Timer 1) signal of Timer 1.
- Finally, the CEN bit of the corresponding (TIMx_CR1 register) must be set to start the two timers separately.

Note: If OCx has been selected as the trigger output of timer 1 (MMS=1xx), its rising edge is used to drive the counter of timer 2.

15.2.12.2. Using one timer to enable another timer

In this example, the enable of timer 2 is controlled by the output comparison of timer 1. Timer 2 counts the divided internal clock only when the OC1REF of Timer 1 is high. The clock frequency of both timers is obtained by dividing CK_INT by 3 (fCK_CNT=fCK_INT/3) by the prescaler.

- Configure Timer 1 as the main mode and send its output comparison reference signal (OC1REF) as the trigger output (MMS=100 of TIM1_CR2 register)
- Configure the OC1REF waveform of timer 1 (TIM1_CCMR1 register)
- Configure timer 2 to get input trigger from timer 1 (TS=000 of TIM2_SMCR register)
- Configure Timer 2 to be gated mode (SMS=101 of TIM2_SMCR register)
- Set CEN=1 of TIM2_CR1 register to enable timer 2
- Set CEN=1 of TIM1_CR1 register to start Timer 1

Note: Timer 2 clock is not synchronized with Timer 1 clock, this mode only affects the enable signal of Timer 2 counter.

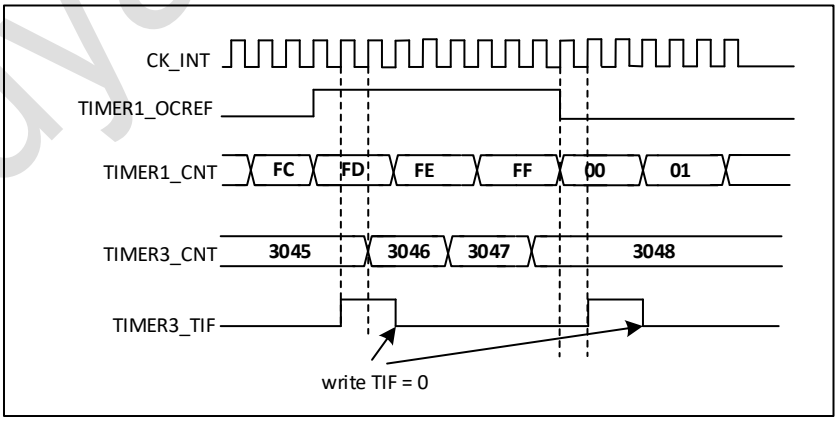


Figure 15-24 OC1REF of timer 1 controls timer 2

Before timer 2 is started, their counters and prescaler are not initialized, so they start counting from the current value. It is possible to reset the 2 timers before starting Timer 1 so that they start from a

given value, i.e. write any value required in the timer counter. The timers can be reset by writing the UG bit of the TIMx_EGR register.

In the next example, it is necessary to synchronize Timer 1 and Timer 2. Timer 1 is the main mode and starts from 0, Timer 2 is the slave mode and starts from 0xE7; both timers have the same prescaler factor. Writing '0' to the CEN bit of TIM1_CR1 will disable Timer 1 and Timer 2 will stop immediately.

- Configure timer 1 as the main mode and send out the output compare 1 reference signal (OC1REF) as the trigger output (MMS=100 of TIM1_CR2 register).
- Configure the OC1REF waveform of timer 1 (TIM1_CCMR1 register).
- Configure timer 2 to get input trigger from timer 1 (TS=000 of TIM2_SMCR register)
- Configure Timer 2 to be gated mode (SMS=101 of TIM2_SMCR register)
- Set UG='1' of TIM1_EGR register to reset timer 1.
- Set UG='1' of TIM2_EGR register to reset timer 2.
- Write '0XE7' to Timer2 counter (TIM2_CNTL) and initialize it to 0xE7.
- Set CEN='1' of TIM2_CR1 register to enable Timer 2.
- Set CEN='1' of TIM1_CR1 register to enable timer 1.
- Set CEN='0' of TIM1_CR1 register to stop Timer 1.

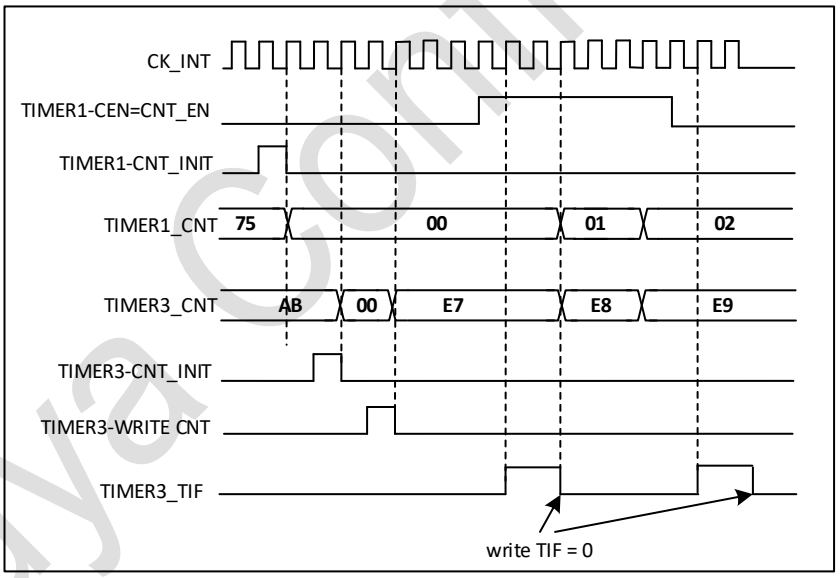


Figure 15-25 Timer 2 can be controlled by enabling Timer 1

15.2.12.3. Using one timer to start another timer

In this example, Timer 2 is enabled using the update event of Timer 1. Once Timer 1 generates an update event, Timer 2 starts counting from its current value (which can be non-zero) according to the divided internal clock. On receipt of a trigger signal, the CEN bit of Timer 2 is automatically set to '1' and the counter starts counting until a '0' is written to the CEN bit of the TIM2_CR1 register. The clock frequency of both timers is divided by the prescaler to CK_INT by 3 (fCK_CNT=fCK_INT/3).

- Configure Timer 1 to be the master mode and send its update event (UEV) as the trigger output (MMS=010 of TIM1_CR2 register).
- Configure the period of timer 1 (TIM1_ARR register).
- Configure timer 2 to get input trigger from timer 1 (TS=000 of TIM2_SMCR register)
- Configure Timer 2 to be in trigger mode (SMS=110 of TIM2_SMCR register)
- Set CEN=1 of TIM1_CR1 register to start Timer 1.

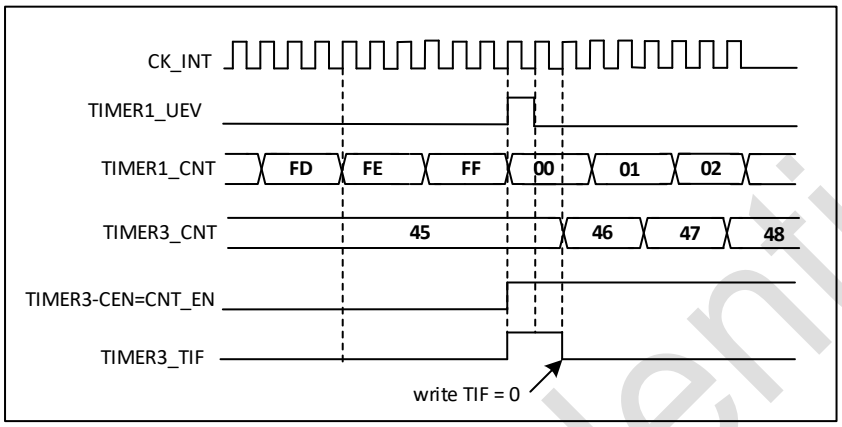


Figure 15-26 Trigger Timer 2 with update of Timer 1

In the previous example, both counters can be initialized before starting counting. Shows the action in the same configuration as 0, using trigger mode instead of gated mode (SMS=110 in TIM2_SMCR register).

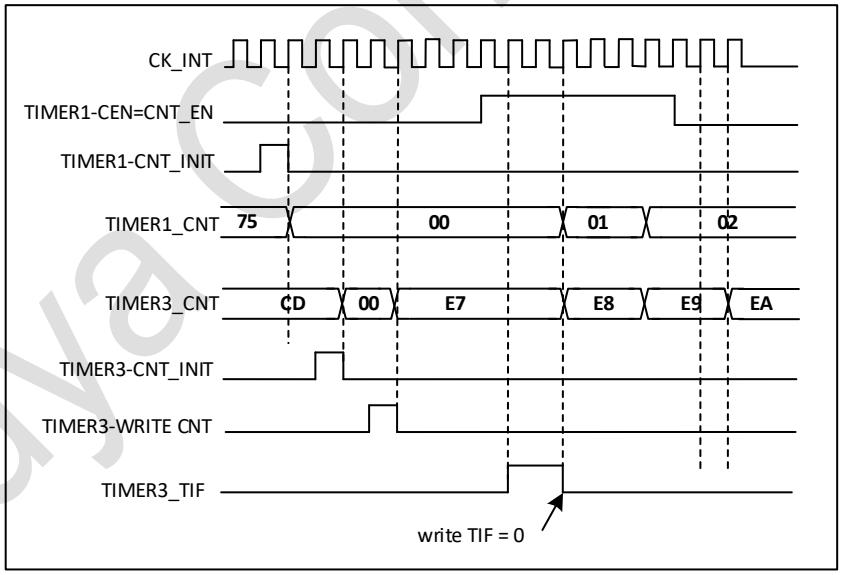


Figure 15-27 Triggering Timer 2 with enable of Timer 1

15.2.12.4. Use one timer as a prescaler for another

This example uses timer 1 as a prescaler for timer 2. The configuration is as follows:

- Configure Timer 1 as the main mode and send its update event UEV as the trigger output (MMS='010' of TIM1_CR2 register). Then output a cycle signal each time the counter overflows;

- Configuring the period of timer 1 (TIM1_ARR register);
- Configuring Timer 2 to get an input trigger from Timer 1 (TS=000 of TIM2_SMCR register);
- Configuring Timer 2 to use external clock mode (SMS=111 of TIM2_SMCR register);
- Setting CEN=1 of TIM1_CR2 register to start Timer 2;
- Set CEN=1 of TIM1_CR1 register to start Timer 1.

15.2.12.5. Using an external trigger start 2 timers synchronously

In this example, Timer 1 is enabled when the TI1 input of Timer 1 rises, and Timer 2 is enabled at the same time as Timer 1. To ensure the alignment of the counter, Timer 1 must be configured in master/slave mode (corresponding to TI1 as slave and Timer 2 as master):

- Configure Timer 1 as master mode and send out its enable as trigger output (MMS=001 of TIM1_CR2 register).
- Configure Timer 1 as slave mode, and get input trigger from TI1 (TS=100 of TIM1_SMCR register).
- Configure Timer 1 as trigger mode (SMS=110 of TIM1_SMCR register).
- Configure Timer 1 in Master/Slave mode with MSM=1 of TIM1_SMCR register.
- Configure Timer 2 to get input triggering from Timer 1 (TS=000 of TIM2_SMCR register)
- Configure Timer 2 for trigger mode (SMS=110 of TIM2_SMCR register).

When a rising edge appears on TI1 of Timer 1, both timers start counting synchronously according to the internal clock and both TIF flags are set at the same time.

Note: In this example, both timers are initialized (set the corresponding UG bits) before starting, and both counters start from 0, but an offset can be inserted between the timers by writing to any of the counter registers (TIMx_CNT). In the figure below you can see a delay between CNT_EN and CK_PSC of timer 1 in master/slave mode.

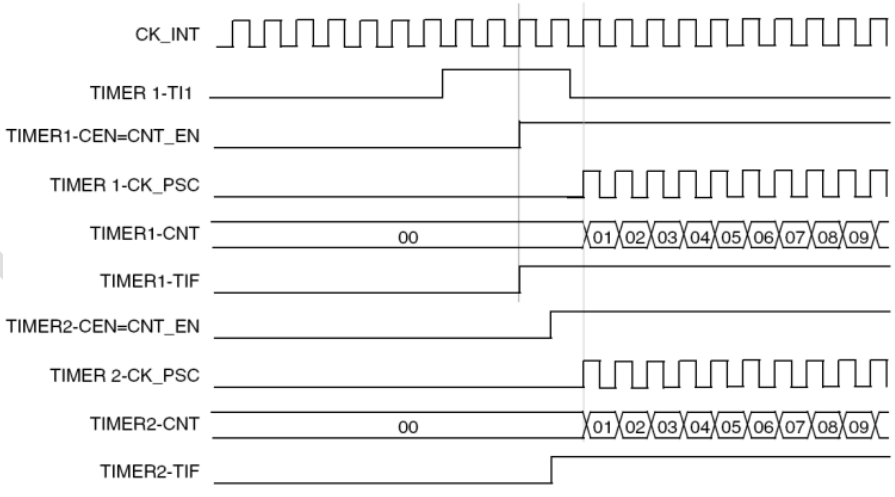


Figure 15-28 Triggering Timer 1 and Timer 2 using TI1 input of Timer 1

15.2.13. Debug mode

When the microcontroller enters debug mode (Cortex-M4 core stopped), the TIMx counter can either continue normal operation or stop, depending on the setting of DBG_TIMx_STOP in the DBG module.

15.3. Register Description

TIM9 register base address: 0x4001 4C00 - 0x4001 4FFF

TIM12 register base address: 0x4000 1800 - 0x4000 1BFF

15.3.1. TIM9 and TIM12 control register 1 (TIMx_CR1)

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
Reserved						RW	RW	RW	RW			RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:10	Reserved, always 0			
9:8	CKD	RW	0	Clock division These 2 bits define the division ratio between the timer clock (CK_INT) frequency, the dead time and the sample clock (tDTS) used by the dead time generator and the digital filter (ETR,TIx). 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: Reserved, do not use this configuration
7	ARPE	RW	0	Auto-reload preload enable bit 0: TIMx_ARR register is not buffered; 1: TIMx_ARR register is buffered.
6:4	Reserved, always 0			
3	OPM	RW	0	One pulse mode 0: the counter does not stop when an update event occurs; 1: The counter stops when the next update event occurs (clearing the CEN bit).
2	URS	RW	0	Update request source The software selects the source of the UEV event by this bit 0: If the update interrupt request is enabled, the update interrupt request is generated by any of the following events: - Counter overflow - Set UG bit - Update generated from the mode controller 1: If the update interrupt request is enabled, only the counter overflow generates an update interrupt request.
1	UDIS	RW	0	Update disable

Bit	Name	R/W	Reset Value	Function
				The software allows/disables the generation of UEV events with this bit Update (UEV) events are generated by any of the following events: - Counter overflow - Set UG bit - Updates generated from the mode controller Registers with cache are loaded with their preloaded values. (Translation: Update shadow registers) 1: UEV is disabled. no update event is generated and the shadow registers (ARR, PSC, CCRx) maintain their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescaler are reinitialized.
0	CEN	RW	0	Counter enable 0: disable the counter; 1: Enable the counter. Note: The external clock and gated mode can only work after the CEN bit is set in software. Trigger mode can automatically set the CEN bit by hardware.

15.3.2. TIM9 and TIM12 Slave Mode Control Register (TIMx_SMCR)

Address offset:0x08

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:7	Reserved, always read as 0.			
6:4	TS	RW	0	Trigger selection These 3 bits are selected for the trigger input of the synchronous counter. 000: Internal trigger 0 (ITR0) 100: Edge detector of TI1 (TI1F_ED) 001: Internal trigger 1 (ITR1) 101: Timer input 1 after filtering (TI1FP1) 010: Internal trigger 2 (ITR2) 110: Filtered timer input 2 (TI2FP2) 011: Internal trigger 3 (ITR3) 111: Reserved Note: These bits can only be changed when they are not used (e.g. SMS=000) to avoid false edge detection when they are changed.
3	Reserved, always read as 0.			
2:0	SMS	RW	0	Slave mode selection When an external signal is selected, the active edge of the trigger signal (TRGI) is related to the polarity of the selected external input (see the description of the input control register and the control register)

Bit	Name	R/W	Reset Value	Function
				000: Slave mode off - if CEN=1, the prescaler is driven directly by the internal clock. 001: Reserved 010: Reserved 011: Reserved 100: Reset mode - The rising edge of the selected trigger input (TRGI) reinitializes the counter and generates a signal to update the registers. 101: Gated Mode - When the trigger input (TRGI) is high, the counter is clocked on. Once the trigger input becomes low, the counter is stopped (but not reset). The counter is started and stopped in a controlled manner. 110: Trigger mode - The counter starts (but does not reset) on the rising edge of the trigger input TRGI; only the counter start is controlled. 111: External Clock Mode 1 - The counter is driven on the rising edge of the selected trigger input (TRGI). Note: Do not use the gated mode if TI1F_EN is selected as the trigger input (TS=100). This is because TI1F_ED outputs a pulse each time TI1F changes, however, the gated mode is to check the level of the trigger input.

Table 15-1 TIMx internal trigger connection

From Timer	ITR0(TS=000)	ITR1(TS=001)	ITR2(TS=010)	ITR3(TS=011)
TIM9	TIM2_TRGO	TIM3_TRGO	TIM10_OC	TIM11_OC
TIM12	TIM4_TRGO	TIM5_TRGO	TIM13_OC	TIM14_OC

15.3.3. TIM9 and TIM12 interrupt enable register (TIMx_DIER)

Address offset:0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TIE	Reserved			CC2IE	CC1IE	UIE
Reserved									RW	Reserved			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:7	Reserved, always 0			
6	TIE	RW	0	Trigger interrupt enable 0: Trigger interrupt is disabled; 1: Trigger interrupt enable.
5:3	Reserved, always 0			
2	CC2IE	RW	0	Capture/Compare 2 interrupt enable 0: disable the Capture/Compare 2 interrupt;

Bit	Name	R/W	Reset Value	Function
				1: Allow Capture/Compare 2 interrupt.
1	CC1IE	RW	0	Capture/Compare 1 interrupt enable 0: disable the Capture/Compare 1 interrupt; 1: Allow Capture/Compare 1 interrupt
0	UIE	RW	0	Update interrupt enable 0: disable the update interrupt; 1: Allow update interrupt.

15.3.4. TIM9 and TIM12 status register (TIMx_SR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
Reserved					RC_W0	RC_W0	Reserved		RC_W0	Reserved			RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
15:11	Reserved, always 0			
10	CC2OF	RC_W0	0	Capture/Compare 2 overcapture flag See CC1OF description.
9	CC1OF	RC_W0	0	Capture/Compare 1 overcapture flag This flag can be set to 1 by hardware only when the corresponding channel is configured for input capture. writing a 0 clears this bit. 0: no duplicate captures are generated; 1: The value of the counter is captured to the TIMx_CCR1 register when the status of CC1IF is already '1'.
8:7	Reserved, always 0			
6	TIF	RC_W0	0	Trigger interrupt flag A hardware '1' for this position when a trigger event occurs (a valid edge is detected at the TRGI input when the slave controller is in a mode other than gated mode, or any edge in gated mode). It is cleared '0' by software. 0: no trigger event is generated; 1: Trigger interrupt waiting for response.
5:3	Reserved, always 0			
2	CC2IF	RC_W0	0	Capture/Compare 2 interrupt flag Refer to CC1IF description.
1	CC1IF	RC_W0	0	Capture/Compare 1 interrupt flag If channel CC1 is configured in output mode:

Bit	Name	R/W	Reset Value	Function
				This bit is set to 1 by hardware when the counter value matches the compare value. it is cleared '0' by software. 0: no match occurs; 1: The value of TIMx_CNT matches the value of TIMx_CCR1. When the content of TIMx_CCR1 is greater than the content of TIMx_APR, the CC1IF bit becomes high in the counter overflow condition If channel CC1 is configured in input mode: This bit is set '1' by hardware when a capture event occurs, it is cleared '0' by software or '0' by reading TIMx_CCR1. 0: no input capture is generated; 1: Counter value has been captured (copied) to TIMx_CCR1 (edge of the same polarity as selected is detected on IC1).
0	UIF	RC_W0	0	Update interrupt flag This bit is set to '1' by hardware when an update event is generated. It is cleared to '0' by software. 0: no update event is generated; 1: Update interrupt waiting for response. This bit is set to '1' by hardware when the register is updated: - If UDIS=0 of TIMx_CR1 register, it is set up when the repeat counter value overflows. - If URS=0 and UDIS=0 of TIMx_CR1 register, the update event is generated when UG=1 of TIMx_EGR register is set, when the counter CNT is reinitialized by software. - If URS=0 and UDIS=0 of TIMx_CR1 register, when the counter CNT is reinitialized by the trigger event (refer to SMCR register description).

15.3.5. TIM9 and TIM12 event generation register (TIMx_EGR)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Reserved			CC2G	CC1G	UG
Reserved									W	Reserved			W	W	W

Bit	Name	R/W	Reset Value	Function
15:7	Reserved, always 0			
6	TG	W	0	Trigger generation This bit is set to '1' by software to generate a trigger event, and is automatically cleared '0' by hardware. 0: no action;

Bit	Name	R/W	Reset Value	Function
				1: TIF=1 of the TIMx_SR register, if the corresponding interrupt is turned on, the corresponding interrupt is generated.
5:3	Reserved, always 0			
2	CC2G	W	0	Generate Capture/Compare 2 generation events Refer to the CC1G description.
1	CC1G	W	0	Capture/Compare 1 generation This bit is set to '1' by software to generate a Capture/Compare event, and is automatically cleared to '0' by hardware. 0: no action; 1: Generates a Capture/Compare event on channel CC1: If channel CC1 is configured as output: Set CC1IF=1 and generate the corresponding interrupt if it is turned on. If channel CC1 is configured as input: The current counter value is captured to the TIMx_CCR1 register; set CC1IF=1, and if the corresponding interrupt is turned on, the corresponding interrupt is generated. If CC1IF is already 1, then set CC1OF=1.
0	UG	W	0	Update generation This bit is set to '1' by software and automatically cleared to '0' by hardware. 0: no action; 1: reinitializes the counter and generates an update event. Note that the prescaler counter is also cleared '0' (but the prescaler coefficient remains unchanged).

15.3.6. TIM9 and TIM12 Capture/Compare Mode Control Register 1 (TIMx_CCMR1)

Address offset:0x18

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				ICIF[3:0]				ICIPSC[1:0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

15.3.6.1. Output comparison mode

Bit	Name	R/W	Reset Value	Function
15	Res.			
14:12	OC2M	RW	0	Output Compare 2 mode
11	OC2PE	RW	0	Output Compare 2 preload enable
10	OC2FE	RW	0	Output Compare 2 fast enable
9:8	CC2S	RW	0	Capture/Compare 2 selection

Bit	Name	R/W	Reset Value	Function
				This bit defines the direction of the channel (input/output), and the selection of the input pin: 00: the CC2 channel is configured as an output; 01: CC2 channel is configured as input, IC2 is mapped on TI2; 10: CC2 channel is configured as an input, with IC2 mapped on TI1; 11: CC2 channel is configured as input and IC2 is mapped on TRC. This mode works only when the internal trigger input is selected (selected by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is off (CC2E=0 in the TIMx_CCER register).
7	Res.			
6:4	OC1M	RW	0	Output Compare 1 mode These 3 bits define the action of the output reference signal OC1REF, which determines the value of OC1. OC1REF is active high, while the active level of OC1 depends on the CC1P bit. 000: Freeze. The comparison between the output comparison register TIMx_CCR1 and the counter TIMx_CNT does not work for OC1REF; 001: Set channel 1 as active level when matching. Force OC1REF high when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 010: Set channel 1 to a null level when mating. Force OC1REF low when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 011: Flip-flop. Flips the level of OC1REF when TIMx_CCR1=TIMx_CNT. 100: Force to invalid level. Force OC1REF to low. 101: Force to valid level. Forces OC1REF to high. 110: PWM mode 1- TIMx_CNT<TIMx_CCR1 when channel 1 is valid level, otherwise it is invalid level. 111: PWM mode 2- TIMx_CNT<TIMx_CCR1 when channel 1 is an invalid level, otherwise it is an active level. Note: In PWM mode 1 or PWM mode 2, the OC1REF level changes only when the comparison result is changed or when switching from freeze mode to PWM mode in the output comparison mode.
3	OC1PE	RW	0	Output Compare 1 preload enable 0: disables the preload function of the TIMx_CCR1 register, the TIMx_CCR1 register can be written at any time, and the newly written value takes effect immediately. 1: enables the preload function of the TIMx_CCR1 register, read/write operations are performed only on the preload register, and the preload value of TIMx_CCR1 is loaded into the current register when the update event comes. Note: Only in single pulse mode (OPM=1 of TIMx_CR1 register), PWM mode can be used without confirming the pre-load register, otherwise its action is not determined.
2	OC1FE	RW	0	Output Compare 1 fast enable

Bit	Name	R/W	Reset Value	Function
				This bit is used to speed up the response of the CC output to a trigger input event. 0: Depending on the value of the counter and CCR1, CC1 operates normally, even if the trigger is on. The minimum delay to activate the CC1 output is 5 clock cycles when there is a valid edge on the input to the flip-flop. 1: The active edge of the input to the flip-flop acts as if a comparison match has occurred. Therefore, OC is set to the comparison level independent of the comparison result. The delay between the active edge of the sampling flip-flop and the CC1 output is reduced to 3 clock cycles. OCFE only works when the channel is configured to PWM1 or PWM2 mode.
1:0	CC1S	RW	0	Capture/Compare 1 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: CC1 channel is configured as an input, with IC1 mapped on TI2; 11: CC1 channel is configured as input and IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (selected by the TS bit of the TIMx_SMCR register). Note: CC1S is writable only when the channel is off (CC1E=0 in the TIMx_CCER register).

15.3.6.2. Input capture mode

Bit	Name	R/W	Reset Value	Function
15:12	IC2F	RW	0	Input capture 2 filter
11:10	IC2PSC	RW	0	Input capture 2 prescaler
9:8	CC2S	RW	0	Capture/Compare 2 selection These 2 bits define the direction of the channel (input/output) and the selection of the input pin: 00: CC2 channel is configured as output; 01: CC2 channel is configured as input, IC2 is mapped on TI2; 10: CC2 channel is configured as an input, with IC2 mapped on TI1; 11: CC2 channel is configured as input and IC2 is mapped on TRC. This mode works only when the internal trigger input is selected (selected by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is off (CC2E=0 in the TIMx_CCER register).
7:4	IC1F	RW	0	Input capture 1 filter These bits define the sampling frequency of the TI1 input and the length of the digital filter. The digital filter consists of an event counter that records The digital filter consists of an event counter that records N events and produces a jump in the output 0000: no filter, sampled at fDTS 1000: sampling frequency fSAMPLING=fDTS/8, N=6

Bit	Name	R/W	Reset Value	Function
				0001: Sampling frequency fSAMPLING=fCK_INT, N=2 1001: Sampling frequency fSAMPLING=fDTS/8, N=8 0010: Sampling frequency fSAMPLING=fCK_INT, N=4 1010: Sampling frequency fSAMPLING=fDTS/16, N=5 0011: Sampling frequency fSAMPLING=fCK_INT, N=8 1011: Sampling frequency fSAMPLING=fDTS/16, N=6 0100: Sampling frequency fSAMPLING=fDTS/2, N=6 1100: Sampling frequency fSAMPLING=fDTS/16, N=8 0101: Sampling frequency fSAMPLING=fDTS/2, N=8 1101: Sampling frequency fSAMPLING=fDTS/32, N=5 0110: Sampling frequency fSAMPLING=fDTS/4, N=6 1110: Sampling frequency fSAMPLING=fDTS/32, N=6 0111: Sampling frequency fSAMPLING=fDTS/4, N=8 1111: Sampling frequency fSAMPLING=fDTS/32, N=8
3:2	IC1PSC	RW	0	Input capture 1 prescaler These 2 bits define the prescaler coefficient for the CC1 input (IC1). Once CC1E=0 (in TIMx_CCER register), the prescaler is reset. 00: no prescaler, one capture triggered for every edge detected on the capture input; 01: capture triggered every 2 events; 10: a capture triggered every 4 events; 11: capture triggered every 8 events.
1:0	CC1S	RW	0	Capture/Compare 1 Selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: the CC1 channel is configured as an output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: CC1 channel is configured as an input, with IC1 mapped on TI2; 11: CC1 channel is configured as input and IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (selected by the TS bit of the TIMx_SMCR register). Note: CC1S is writable only when the channel is off (CC1E=0 in the TIMx_CCER register).

15.3.7. TIM9 and TIM12 Capture/Compare Enable Registers (TIMx_CCER)

Address offset:0x20

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
Reserved								RW	Reserved	RW	RW	RW	Reserved	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	Reserved, always 0			
7	CC2NP	RW	0	Capture/Compare 2 output polarity Refer to the description of CC1P.
6	Reserved, always 0			
5	CC2P	RW	0	Capture/Compare 2 output polarity Refer to the description of CC1P.
4	CC2E	RW	0	Capture/Compare 2 output enable Refer to the description of CC1E.
3	CC1NP	RW	0	Capture/Compare 1 complementary output polarity When the CC1 channel is configured as an output: CC1NP must be held at 0; When CC1 channel is configured as input: CC1NP is used in conjunction with CC1P to act on TIxFP1 polarity (refer to CC1P description)
2	Reserved, always 0			
1	CC1P	RW	0	Capture/Compare 1 output polarity CC1 channel configured as output: 0: OC1 active high; 1: OC1 low active. CC1 channel is configured as input: The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 as trigger or capture signals. 00: Non-inverting/rising edge: TIxFP1 rising edge active (capture, trigger in reset mode, external clock or in trigger mode); TIxFP1 not inverted (gated mode). 01: Inverted/falling edge: TIxFP1 falling edge valid (capture, trigger in reset mode, external clock or trigger mode); TIxFP1 inverted (gated mode). 10: Reserved, do not use this configuration. 11: Not inverted/dual edge Both rising and falling edges of TIxFP1 are active (capture, triggered in reset mode, external clock or trigger mode); TIxFP1 not inverted (gated mode).
0	CC1E	RW	0	Capture/Compare 1 output enable CC1 channel configured as output: 0: Turn off - OC1 disable output. 1: Enables - OC1 signal output to the corresponding output pin. CC1 channel configured as input: This bit determines whether the counter value can be captured into the TIMx_CCR1 register. 0: Capture disable; 1: Capture enable.

Table 15-2 Control bits for output channel OCx and

CCxE bit	OCx output status
0	Output disable (Disconnected from timer) OCx=0. OCx_EN=0
1	OCx=OCxREF+Polarity, OCx_EN=1

Note: The status of the external I/O pins connected to the complementary OCx channels depends on the OCx channel status and the GPIO registers.

15.3.8. TIM9 and TIM12 Counters (TIMx_CNT)

Address offset:0x24

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	Counter value

15.3.9. TIM9 and TIM12 Prescaler (TIMx_PSC)

Address offset:0x28

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	PSC	RW	0	Prescaler value The clock frequency of the counter (CK_CNT) is equal to fCK_PSC/(PSC[15:0]+1). PSC contains the value loaded into the current prescaler register each time an update event is generated; the update event consists of the counter being The update event includes the counter being cleared '0' by the UG bit of TIM_EGR or by a slave controller operating in reset mode.

15.3.10. TIM9 and TIM12 Auto Reload Register (TIMx_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	Prescaler value The ARR contains the value that will be loaded into the actual autoreload register. The counter does not work when the autoreload value is empty.

15.3.11. TIM9 and TIM12 Capture/Compare Register 1 (TIMx_CCR1)

Address offset:0x34

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
RW															

Bit	Name	R/W	Reset Value	Function
15:0	CCR1	RW	0	Capture/Compare 1 value If the CC1 channel is configured as an output: CCR1 contains the value loaded into the current Capture/Compare 1 register (Preload value). If the preload function is not selected in the TIMx_CCMR1 register (OC1PE bit), the written value is immediately transferred to the current register. Otherwise, this preload value is transferred to the current capture/compare 1 register only when an update event occurs. The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates an output signal on the OC1 port. If the CC1 channel is configured as input: CCR1 contains the counter value transferred from the last input capture 1 event (IC1).

15.3.12. TIM9 and TIM12 Capture/Compare Register 2 (TIMx_CCR2)

Address offset:0x38

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2															

RW

Bit	Name	R/W	Reset Value	Function
15:0	CCR2	RW	0	Capture/Compare 2 value If the CC2 channel is configured as an output: CCR2 contains the value loaded into the current Capture/Compare 2 register (Preload value). If the preload function is not selected in the TIMx_CCMR2 register (OC2PE bit), the written value is immediately transferred to the current register. Otherwise, this preload value is transferred to the current capture/compare2 register only when an update event occurs. The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates an output signal on the OC2 port. If the CC2 channel is configured as input: CCR2 contains the counter value transferred from the last input capture 2 event (IC2).

15.3.13. TIM9 and TIM12 register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD		ARPE	Reserved			OPM	URS	UDIS	CEN
	Read/Write																							rw		rw				rw	rw	rw	rw
	Reset Value																							0		0				0	0	0	0
0x08	TIMx_SMCR	Reserved																									TS			Reserved	SMS		
	Read/Write																										rw				rw		
	Rese																										0				0		

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	tValue																																
0x0C	TImx_DIER	Reserved																									TIE	Reserved			CC2IE	CC1IE	UIE
	Read/Write																										rw				rw	rw	rw
	Reset Value																										0				0	0	0
0x10	TImx_SR	Reserved																					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
	Read/Write																						rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0
	Reset Value																						0	0			0				0	0	0
0x14	TImx_EGR	Reserved																									TG	Reserved			CC2G	CC1G	UG
	Read/Write																										w				w	w	w
	Reset Value																										0				0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	TImx_CCMR1	Reserved																Reserved	OC2M			OC2PE	OC2FE	CC2S		Reserved	OC1M			OC1PE	OC1FE	CC1S	
																			IC2F			IC2PSC					IC1F			IC1PSC			
	Read/Write																	rw															
	Reset Value																	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
0x20	TImx_CCER	Reserved																								CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Read/Write																									rw		rw	rw	rw		rw	rw
	Reset Value																									0		0	0	0		0	0
0x24	TImx_CNT	Reserved																CNT															
	Read/Write																	rw															
	Reset Va																	0															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	lue																																
0x28	TIMx_PSC	Reserved																PSC															
	Read/Write																	rw															
	Reset Value																	0															
0x2C	TIMx_ARR	Reserved																ARR															
	Read/Write																	rw															
	Reset Value																	0xFFFF															
0x34	TIMx_CCR1	Reserved																CCR1															
	Read/Write																	rw/ro															
	Reset Value																	0															
0x	TIMx	Reserved																															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
38	_CCR2																																
	Read/Write																	rw/ro															
	Reset Value																	0															

16. General-purpose timer(TIM10/TIM11/TIM13/TIM14)

16.1. Introduction

The general purpose timer (TIM10/TIM11/TIM13/TIM14, Hereafter referred to as TIMx) consists of a 16-bit auto-load counter driven by a programmable prescaler.

It is suitable for a variety of uses, including measuring the pulse width of an input signal (input capture), or generating an output waveform (output comparison, PWM). Using the timer prescaler and RCC clock control prescaler, pulse width and waveform period can be adjusted from a few microseconds to a few milliseconds.

The general-purpose timers (TIMx) are completely independent; they do not share any resources. They can operate synchronously.

16.1.1. TIMx main features

The TIMx timer features include:

- 16-bit upward automatic counter loading
- 16-bit programmable (modifiable in real time) prescaler with a dividing factor of the counter clock frequency of any value between 1 and 65536
- 2 independent channel:
  - Input capture
  - Output comparison
  - PWM generation (edge mode)
  - Single pulse mode output
- Synchronization circuits to control timer-to-timer interconnections via external signals
- Interrupt generation when the following events occur:
  - Update: counter up overflow, counter initialization (Triggered by software or internally)
  - Trigger event (counter starts, stops, initializes, or is internally triggered to count)
  - Input capture
  - Output comparison

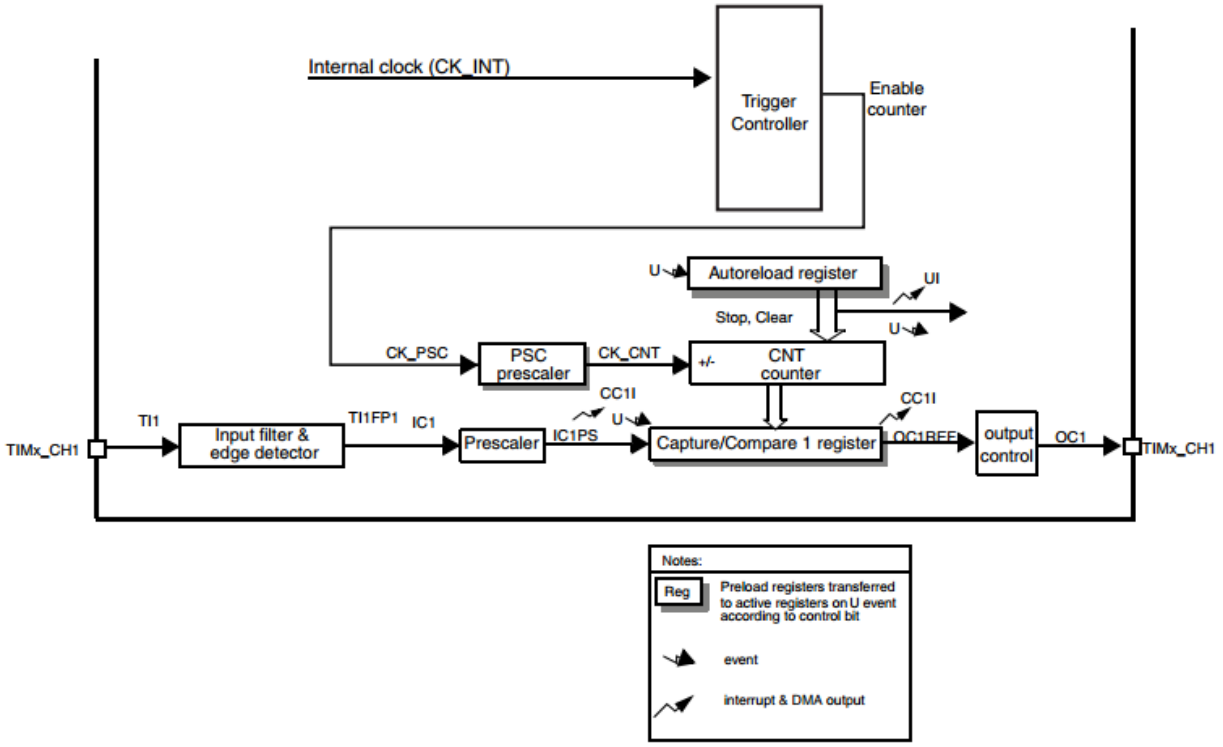


Figure 16-1 TIMx(TIM10/TIM11/TIM13/TIM14) module

16.2. TIMx functional description

16.2.1. Time-base unit

The main part of the general purpose timer is a 16-bit counter and its associated auto-load register. This counter counts up.

The clock of the counter can be divided by a prescaler.

The counter, the autoloader register and the prescaler register can be read and written by software, even if the counter is still running.

The time base unit contains:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoload register (TIMx_ARR)

The Autoload register is preloaded and writing or reading the Autoreload register will access its preloaded register. Depending on the setting of the Auto Reload Preload Enable bit (ARPE) in the TIMx_CR1 register, the contents of the preload register are transferred to the shadow register immediately or at each update event (UEV). An update event is generated when the counter reaches the overflow condition and when the UDIS bit in the TIMx_CR1 register equals 0. Update events can also be generated by software and other conditions. The generation of update events in each configuration will be described in detail later.

The counter is driven by the prescaler divided clock output CK_CNT, which is valid for the counter only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the counter starts counting one clock cycle after the CEN bit of the TIMx_CR register is set.

16.2.1.1. Prescaler description

The prescaler can divide the counter clock frequency by any value between 1 and 65536. It is based on a 16-bit counter controlled by a 16-bit register (in the TIMx_PSC register). Because this control register has a buffer, it is able to be changed at runtime. The new prescaler parameters will be adopted when the next update event comes.

An example of changing the counter parameters while the prescaler is running is shown below.

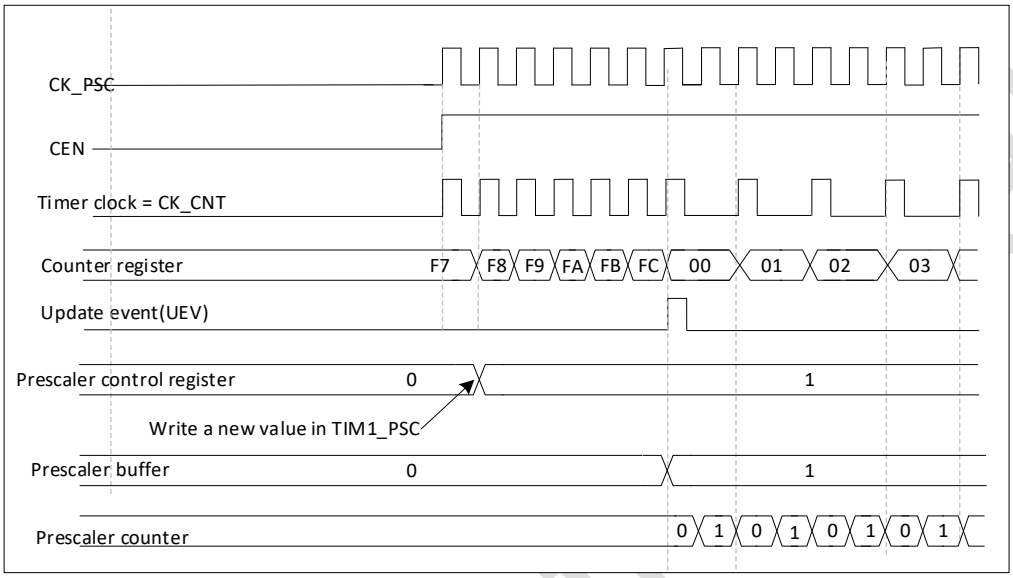


Figure 16-2 Counter timing diagram with prescaler division change from 1 to 2

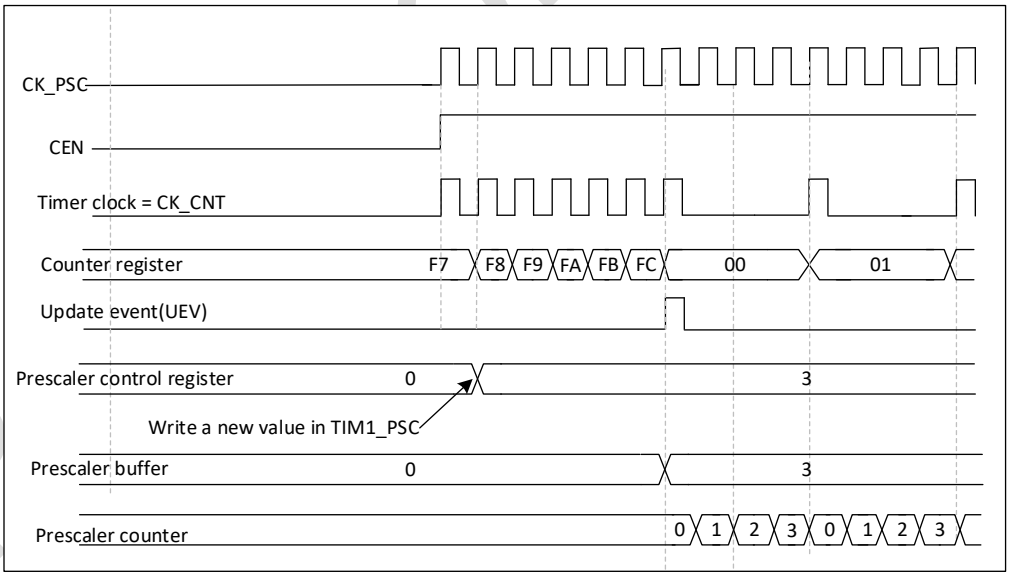


Figure 16-3 Counter timing diagram with prescaler division change from 1 to 4

16.2.2. Counter Mode

16.2.2.1. Upcounter mode

In up-count mode, the counter counts from 0 to the auto-load value (the contents of TIMx_ARR), then starts counting from 0 again and generates a count overflow event.

Setting the UG bit in the TIMx_EGR register (by software) can also generate an update event.

The update event can be disabled by setting the UDIS bit in the TIMx_CR1 register; this prevents the shadow register from being updated when a new value is written to the preload register. An update event will not be generated until the UDIS bit is cleared '0'. Even then, the counter will still be cleared '0' when an update event should be generated, and the internal counter of the prescaler is also cleared '0' (but the value of the prescaler remains unchanged).

In addition, if the URS bit (Select Update Request Source) in the TIMx_CR1 register is set, an update event UEV can be generated by setting the UG bit, but the UIF flag bit will not be set up (i.e., no interrupt request will be generated). This is to avoid clearing the counter on a capture event and generating both an update and a capture interrupt.

When an update event occurs, all of the following registers are updated and the hardware sets the update flag bit (UIF bit in the TIMx_SR register) simultaneously (based on the URS bit):

- The autoload shadow register is reset to the value of the preload register (TIMx_ARR).
- The prescaler buffer is reset to the value of the preload register (the contents of the TIMx_PSC register).

The following figure gives some examples of how the counter acts at different clock frequencies when TIMx_ARR=0x36.

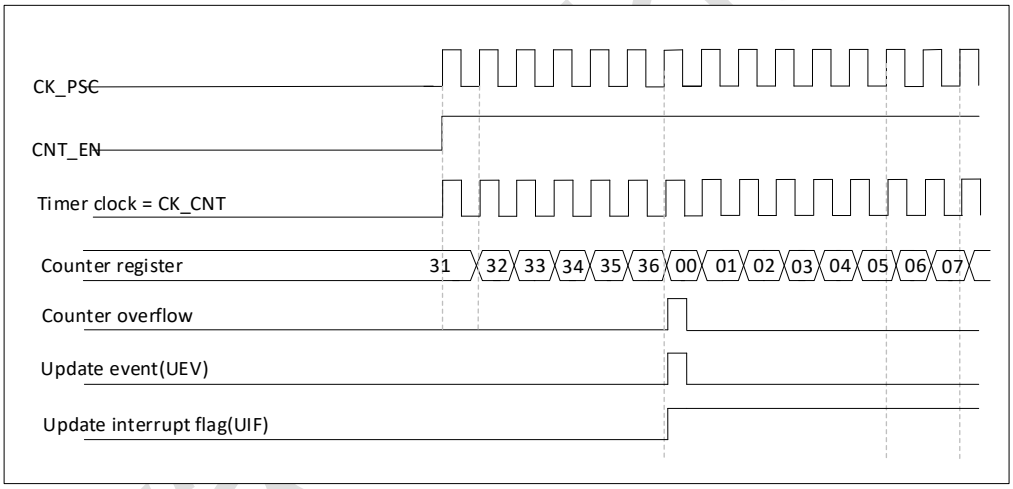


Figure 16-4 Counter timing diagram, internal clock divided by 1

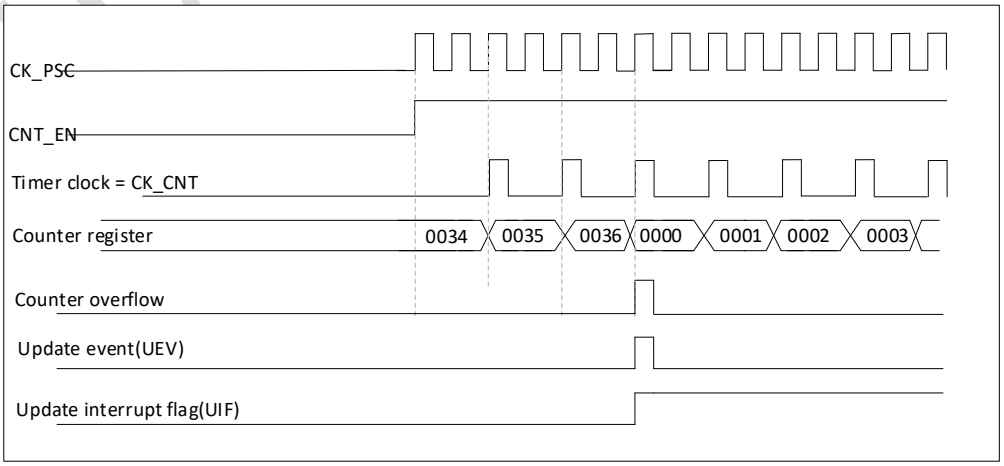


Figure 16-5 Counter timing diagram, internal clock divided by 2

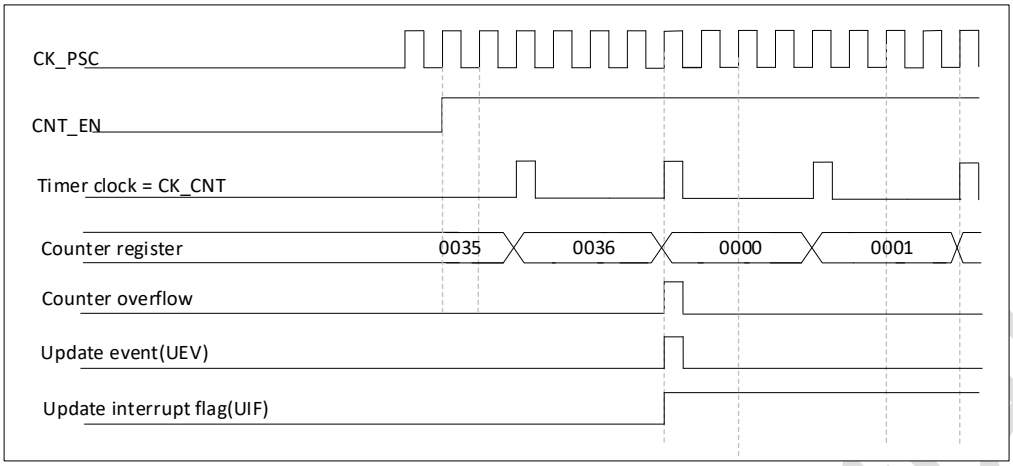


Figure 16-6 Counter timing diagram, internal clock divided by 4

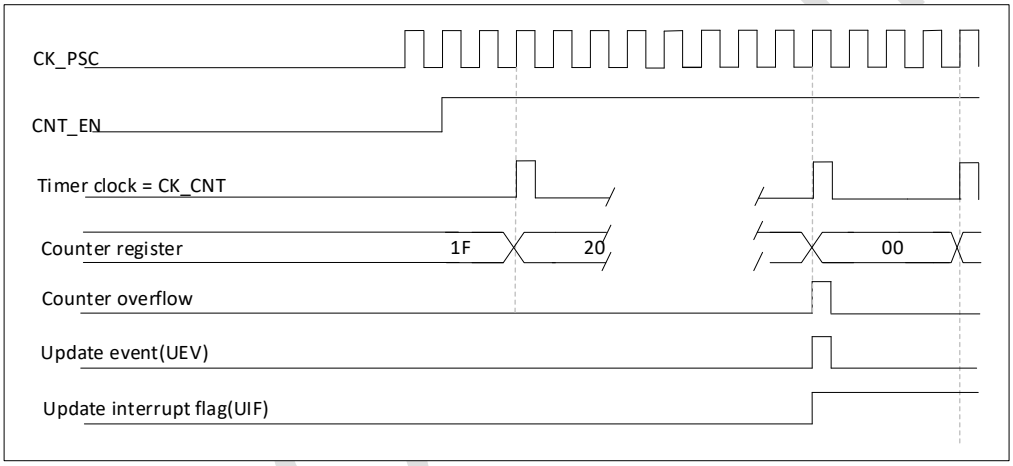


Figure 16-7 Counter timing diagram, internal clock divided by N

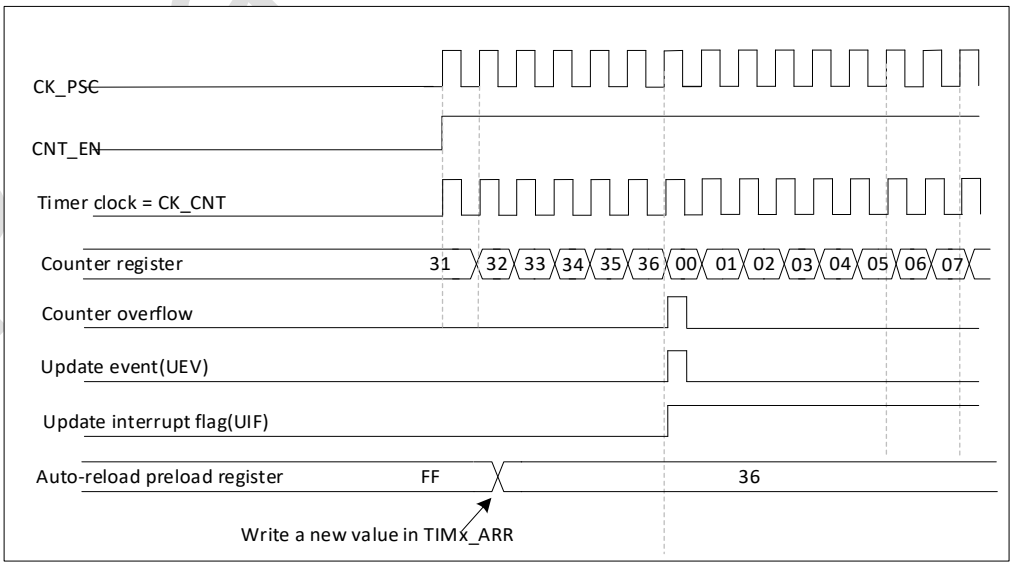


Figure 16-8 Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

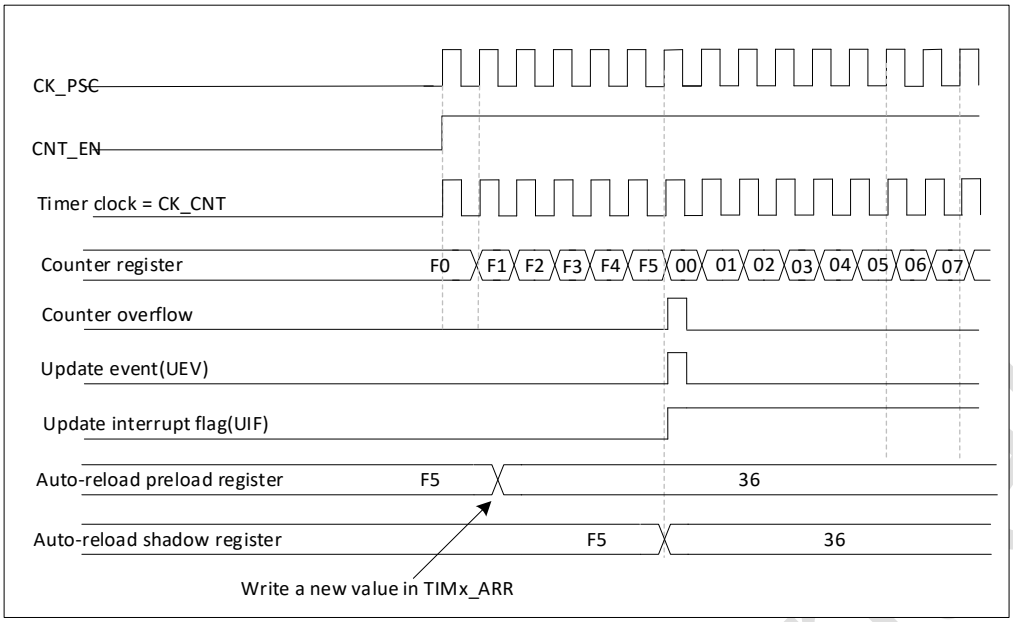


Figure 16-9 Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)

16.2.3. Clock Selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)

16.2.3.1. Internal clock source (CK_INT)

For TIM10/TIM11/TIM13/TIM14, the CEN and UG bits (TIMx_EGR register) are the de facto control bits and can only be modified by software (the UG bit is still automatically cleared) when the internal clock source is the default clock. As long as the CEN bit is written to '1', the prescaler clock is provided by the internal clock CK_INT.

The following diagram shows the operation of the control circuit and the up counter in general mode without the prescaler.

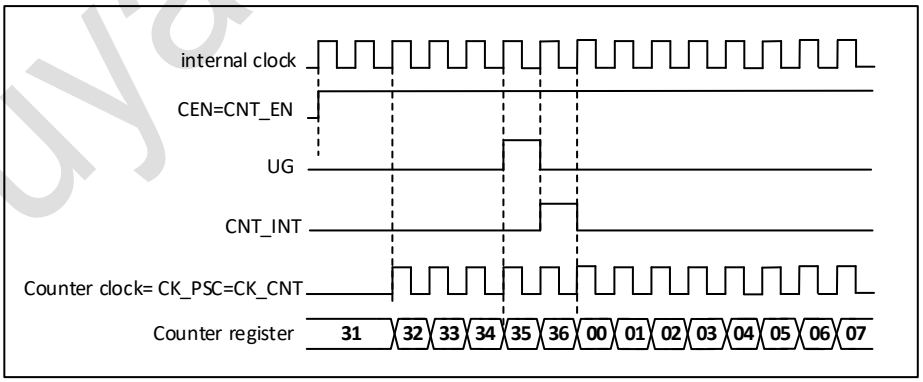


Figure 16-10 Control circuit in normal mode, internal clock divided by 1

16.2.4. Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

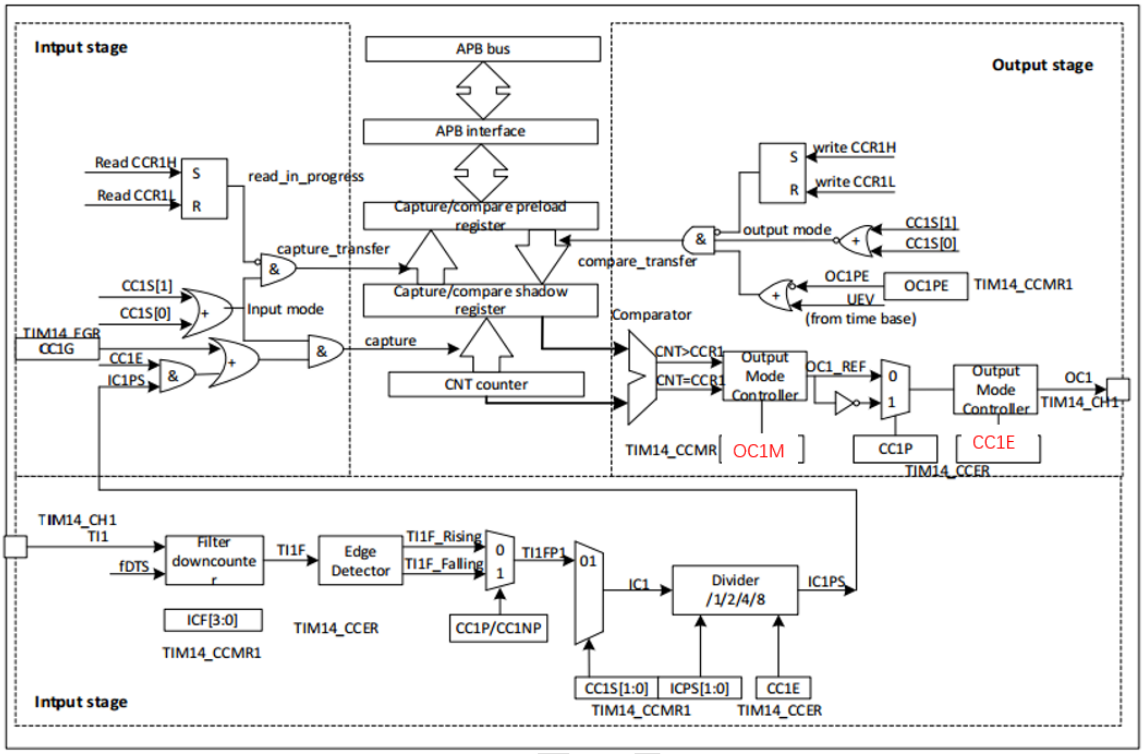


Figure 16-11 Capture/compare channel

The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.The capture/compare block is made of one preload register and one shadow register. Write and read only access the preload register.In capture mode, captures are actually done in the shadow register, which is copied into the preload register.In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter

16.2.5. Input capture mode

In the input capture mode, the current value of the counter is latched into the capture/compare register (TIMx_CCRx) when the corresponding edge on the ICx signal is detected. When a capture event occurs, the corresponding CCxIF flag (TIMx_SR register) is set to 1. If an interrupt operation is open, an interrupt request will be generated. If the CCxIF flag is already high when a capture event occurs, the over-capture flag CCxOF (TIMx_SR register) is set to 1. CCxIF can be cleared by writing CCxIF=0, or by reading the capture data stored in the TIMx_CCRx register. CCxOF can be cleared by writing CCxOF=0.

The following example shows how to capture the counter value into the TIMx_CCMR1 register on the rising edge of the TI1 input, as follows:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to '01' in the TIMx_CCMR1 register. As soon as CC1S becomes different from '00', the channel is configured in input mode and the TIMx_CCR1 register becomes readonly.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). When toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to '0011' in the TIMx_CCMR1 register.
- Select the valid conversion edge of TI1 channel and write CC1P and CC1NP bits to 00 (set to rising edge) in the TIMx_CCER register.
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register. When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- If the CC1IE bit is set, an interrupt is generated.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

16.2.6. Forced output mode

In the output mode (CCxS=00 in the TIMx_CCMRx register), the output compare signal (OCxREF and the corresponding OCx/OCxN) can be directly forced by software to the valid or invalid state, regardless of the comparison result between the output compare register and the counter.

The output comparison signal (OCxREF/OCx) can be forced to be valid by setting the corresponding OCxM=101 in the TIMx_CCMRx register. In this way, OCxREF is set high (OCxREF is always active high), and OCx gets a signal of opposite polarity of CCxP.

For example: CCxP=0 (OCx is active high), then OCx is set strongly high.

Setting OCxM=100 in the TIMx_CCMRx register can force the OCxREF signal low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter is still going on, and the corresponding flags are modified. Therefore the corresponding interrupt request will still be generated. This will be described in the Output Compare Mode section below.

16.2.7. Output compare mode

This function is used to control an output waveform or to indicate when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = ’000’), be set active (OCxM = ’001’), be set inactive (OCxM = ’010’) or can toggle (OCxM = ’011’) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx_DIER register).

You can select whether the TIMx_CCRx register needs to use a preload register by configuring the OCxPE bit in TIMx_CCMRx.

In output compare mode, the update event UEV has no effect on the OCxREF and OCx outputs. The synchronization can be done with an accuracy of one count cycle of the counter. The output compare mode (in single pulse mode) can also be used to output a single pulse.

Configuration steps for output comparison mode:

1. Select the counter clock (internal, external, prescaler).
2. Write the corresponding data into the TIMx_ARR and TIMx_CCRx registers.
3. If an interrupt request is to be generated, set the CCxIE bit.
4. Select the output mode, for example:
   - Flip the output pin of OCx when the counter is requested to match CCRx, set OCxM = 011
   - Set OCxPE = 0 to disable the preload register
   - Set CCxP = 0 to select the polarity to active high
   - Set CCxE = 1 to enable output
5. Set the CEN bit of TIMx_CR1 register to start the counter

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not used (OCxPE='0', otherwise the shadow register of TIMx_CCRx can only be updated when the next update event occurs). An example is given in the following figure.

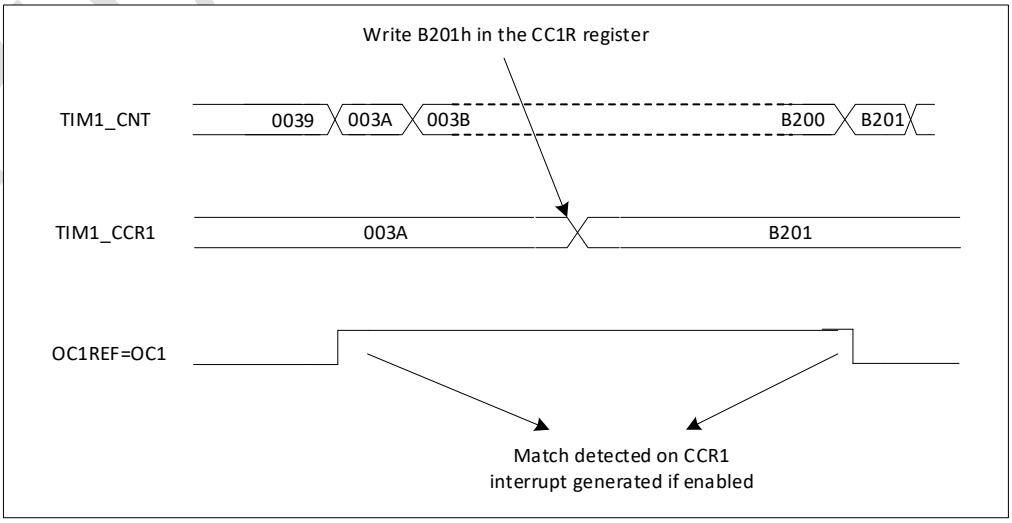


Figure 16-12 Output compare mode, toggle on OC1

16.2.8. PWM mode

The pulse width modulation mode can generate a signal with a frequency determined by the TIMx_ARR register and a duty cycle determined by the TIMx_CCRx register.

The OCxM bit in the TIMx_CCMRx register is written to '110' (PWM mode 1) or '111' (PWM mode 2) to be able to independently set each OCx output channel to generate one PWM. it must be enabled by setting the The corresponding preload registers must be enabled by setting the OCxPE bit of the TIMx_CCMRx register and finally by setting the ARPE bit of the TIMx_CR1 register to enable the preload registers for automatic reloading (in upcount or centrosymmetric mode).

The preload registers can be transferred to the shadow registers only when an update event occurs, so the user must initialize all registers by setting the UG bit in the TIMx_EGR register before the counter starts counting.

The polarity of OCx can be set by software through the CCxP bit in the TIMx_CCER register, which can be set to active high or active low. the output enable of OCx is controlled through the CCxE bit (in TIMx_CCER). See the description of TIMx_CCER register for details.

In PWM mode (mode 1 or mode 2), TIMx_CNT and TIMx_CCRx are always being compared, (based on the counter count direction) to determine if TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx is met.

The timer is capable of generating edge-aligned PWM signals.

16.2.8.1. PWM edge-aligned mode

- Upward counting configuration

  Performs upward counting when the DIR bit in the TIMx_CR1 register is low.

  The following is an example of PWM mode 1. When TIMx_CNT<TIMx_CCRx, PWM reference signal OCxREF is high, otherwise it is low. If the comparison value in TIMx_CCRx is greater than the auto reload value (TIMx_ARR), OCxREF remains '1'. If the comparison value is 0, OCxREF is kept as '0'. The following figure shows an example of edge-aligned PWM waveform when TIMx_ARR=8.

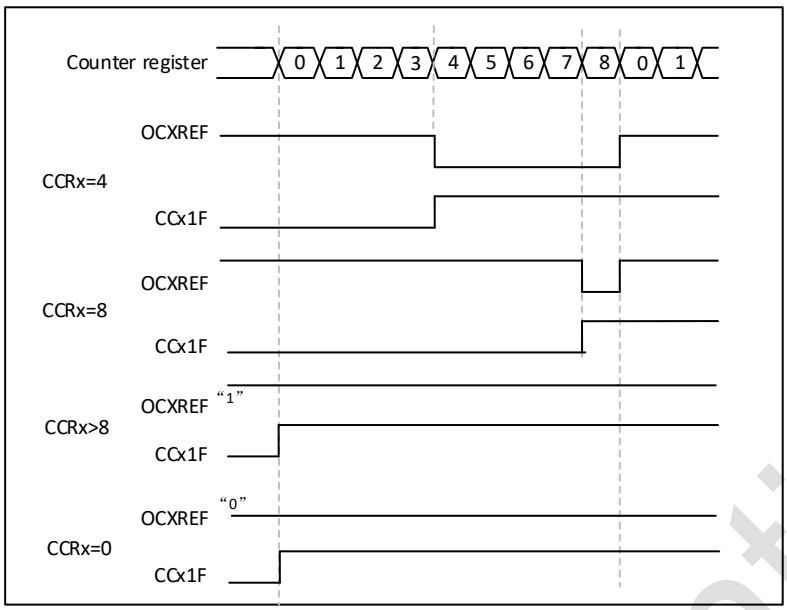


Figure 16-13 Edge-aligned PWM waveforms (ARR = 8)

16.2.9. Single pulse mode

The single pulse mode (OPM) is a special case of the many modes described earlier. This mode allows the counter to respond to an excitation and generate a pulse with a programmable pulse width after a programmable delay.

The counter can be started from the mode controller to generate waveforms in the output compare mode or PWM mode. Setting the OPM bit in the TIMx_CR1 register will select the single pulse mode, which allows the counter to automatically stop when the next update event UEV is generated.

A pulse can be generated only when the comparison value is different from the initial value of the counter. Before starting (when the timer is waiting to be triggered), it must be configured as follows:

- Up counting method: counter CNT < CCRx ≤ ARR (In particular, 0 < CCRx).

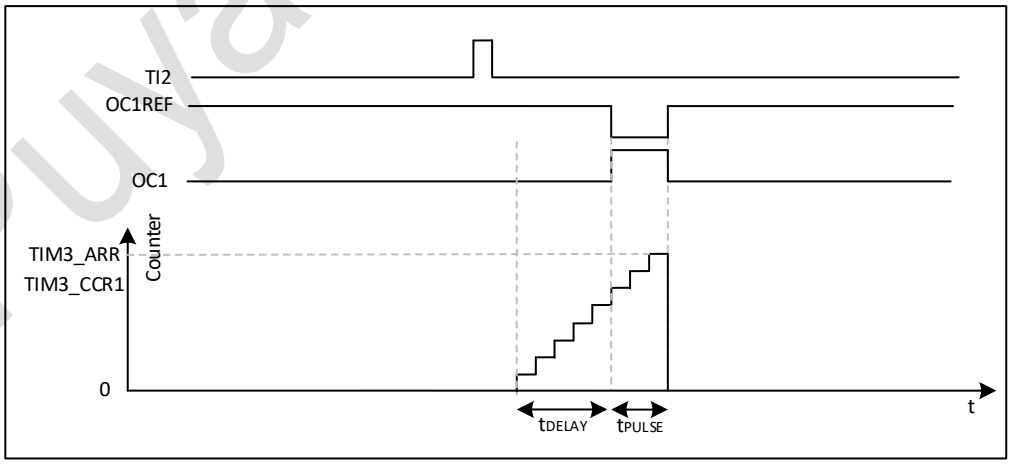


Figure 16-14 Example of single pulse mode

16.2.10. Timer synchronization

All TIMx timers are internally connected for timer synchronization or linking. When one timer is in master mode, it can reset, start, stop or provide clock to the counter of another timer in slave mode. tim10/11/13/14 are used as master timer output oc to the slave timer TIM9/12 under the corresponding configuration.

16.2.11. Debug mode

When the microcontroller is in debug mode (Cortex-M4 core stopped), depending on the setting of DBG_TIMx_STOP in the DBG module, the

TIMx counter can either continue its normal operation or stop.

16.3. Register Description

TIM10 Register base address: 0x4001 5000 - 0x4001 53FF

TIM11 Register base address: 0x4001 5400 - 0x4001 57FF

TIM13 Register base address: 0x4000 1C00 - 0x4000 1FFF

TIM14 Register base address: 0x4000 2000 - 0x4000 23FF

16.3.1. TIMx control register 1 (TIMx_CR1)

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
Reserved						RW	RW	RW	Reserved			RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:10	Reserved, always 0			
9:8	CKD	RW	0	Clock division These 2 bits define the division ratio between the timer clock (CK_INT) frequency, the dead time and the sample clock (tDTS) used by the dead time generator and the digital filter (ETR,TIx). 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: Reserved, do not use this configuration
7	ARPE	RW	0	Auto-reload preload enable bit 0: TIMx_ARR register is not buffered; 1: TIMx_ARR register is buffered.
6:4	Reserved, always 0			
3	OPM	RW	0	One pulse mode 0: the counter does not stop when an update event occurs; 1: The counter stops when the next update event occurs (clearing the CEN bit).

Bit	Name	R/W	Reset Value	Function
2	URS	RW	0	Update request source The software selects the source of the UEV event by this bit 0: If the update interrupt request is enabled, the update interrupt request is generated by any of the following events: - Counter overflow - Set the UG bit 1: If the update interrupt request is enabled, only the counter overflow/underflow will generate the update interrupt request.
1	UDIS	RW	0	Update disable The software allows/disables the generation of UEV events with this bit Update (UEV) events are generated by any of the following events: - Counter overflow - Set UG bit Registers with cache are loaded with their preloaded values. (Translation: Update shadow registers) No update event is generated and the shadow registers (ARR, PSC, CCRx) maintain their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescaler are reinitialized.
0	CEN	RW	0	Counter enable 0: disable the counter; 1: Enable the counter.

16.3.2. TIMx interrupt enable register (TIMx_DIER)

Address offset:0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
Reserved														RW	RW

Bit	Name	R/W	Reset Value	Function
15:2	Reserved, always 0			
1	CC1IE	RW	0	Capture/Compare 1 interrupt enable 0: disable the Capture/Compare 1 interrupt; 1: Allow Capture/Compare 1 interrupt
0	UIE	RW	0	Update interrupt enable 0: disable the update interrupt; 1: Allow update interrupt.

16.3.3. TIMx Status Register (TIMx_SR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CC1OF	Reserved							CC1IF	UIF
Reserved						RC_W0	Reserved							RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
15:10	Reserved, always 0			
9	CC1OF	RC_W0	0	Capture/Compare 1 overcapture flag This flag can be set to 1 by hardware only when the corresponding channel is configured for input capture. writing a 0 clears this bit. 0: no duplicate captures are generated; 1: The value of the counter is captured to the TIMx_CCR1 register when the status of CC1IF is already '1'.
8:2	Reserved, always 0			
1	CC1IF	RC_W0	0	Capture/Compare 1 interrupt flag If channel CC1 is configured in output mode: This bit is set to 1 by hardware when the counter value matches the compare value. it is cleared '0' by software. 0: no match occurs; 1: The value of TIMx_CNT matches the value of TIMx_CCR1. When the content of TIMx_CCR1 is greater than the content of TIMx_APR, the CC1IF bit becomes high on counter overflow If channel CC1 is configured in input mode: This bit is set to '1' by hardware when a capture event occurs, it is cleared '0' by software or by reading TIMx_CCR1 to clear '0'. 0: no input capture is generated; 1: Counter value has been captured (copied) to TIMx_CCR1 (edge of same polarity as selected is detected on IC1).
0	UIF	RC_W0	0	Update interrupt flag This bit is set to '1' by hardware when an update event is generated. It is cleared to '0' by software. 0: no update event is generated; 1: Update interrupt waiting for response. This bit is set to '1' by hardware when the register is updated: - If UDIS=0 of TIMx_CR1 register, when the repeat counter value overflows. - If URS=0 and UDIS=0 of TIMx_CR1 register, when the update event is generated when UG=1 of TIMx_EGR register is set, when the counter CNT is reinitialized by software.

16.3.4. TIMx Event Generation Register (TIMx_EGR)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1G	UG
Reserved														W	W

Bit	Name	R/W	Reset Value	Function
15:2	Reserved, always 0			
1	CC1G	W	0	Capture/Compare 1 generation This bit is set to '1' by software to generate a Capture/Compare event, and is automatically cleared to '0' by hardware. 0: no action; 1: Generates a Capture/Compare event on channel CC1: If channel CC1 is configured as output: Set CC1IF=1 and generate the corresponding interrupt if it is turned on. If channel CC1 is configured as input: The current counter value is captured to the TIMx_CCR1 register; set CC1IF=1, and if the corresponding interrupt is turned on, the corresponding interrupt is generated. If CC1IF is already 1, then set CC1OF=1.
0	UG	W	0	Update generation This bit is set to '1' by software and automatically cleared to '0' by hardware. 0: no action; 1: reinitializes the counter and generates an update event. Note that the prescaler counter is also cleared '0' (but the prescaler coefficient remains unchanged).

16.3.5. TIMx Capture/Compare Mode Control Register 1 (TIMx_CCMR1)

Address offset:0x18

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									OC1M[2:0]			OC1PE	Reserved	CC1S[1:0]	
Reserved								ICIF[3:0]				ICIPSC[1:0]			
Reserved								RW	RW	RW	RW	RW	RW	RW	RW

16.3.5.1. Output comparison mode

Bit	Name	R/W	Reset Value	Function
15：7	Reserved, always 0			

Bit	Name	R/W	Reset Value	Function
6:4	OC1M	RW	0	Output Compare 1 mode These 3 bits define the action of the output reference signal OC1REF, which determines the value of OC1. OC1REF is active high, while the active level of OC1 depends on the CC1P bit. 000: Freeze. The comparison between the output comparison register TIMx_CCR1 and the counter TIMx_CNT does not work for OC1REF; 001: Set channel 1 as active level when matching. Force OC1REF high when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 010: Set channel 1 to a null level when mating. Force OC1REF low when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 011: Flip-flop. Flips the level of OC1REF when TIMx_CCR1=TIMx_CNT. 100: Force to invalid level. Force OC1REF to low. 101: Force to valid level. Force OC1REF to high. 110: PWM mode 1-TIMx_CNT<TIMx_CCR1 when channel 1 is valid level, otherwise, it is invalid level. 111: PWM mode 2-TIMx_CNT<TIMx_CCR1 channel 1 is invalid level, otherwise it is valid level. Note: In PWM mode 1 or PWM mode 2, the OC1REF level changes only when the comparison result is changed or when switching from freeze mode to PWM mode in the output comparison mode.
3	OC1PE	RW	0	Output Compare 1 preload enable 0: disables the preload function of the TIMx_CCR1 register, the TIMx_CCR1 register can be written at any time, and the newly written value takes effect immediately. 1: enables the preload function of the TIMx_CCR1 register, read/write operations are performed only on the preload register, and the preload value of TIMx_CCR1 is loaded into the current register when the update event comes. Note: Only in single pulse mode (OPM=1 of TIMx_CR1 register), PWM mode can be used without confirming the pre-load register, otherwise its action is not determined.
2	Reserved, always 0			
1:0	CC1S	RW	0	Capture/Compare 1 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: Reserved; 11: Reserved; Note: CC1S is writable only when the channel is off (CC1E=0 of TIMx_CCER register).

16.3.5.2. Input capture mode

Bit	Name	R/W	Reset Value	Function
15:8	Reserved, always 0			
7:4	IC1F	RW	0	Input capture 1 filter These bits define the sampling frequency of the TI1 input and the length of the digital filter. The digital filter consists of an event counter that records The digital filter consists of an event counter that records N events and produces a jump in the output 0000: no filter, sampled at fDTS 1000: sampling frequency fSAMPLING=fDTS/8, N=6 0001: Sampling frequency fSAMPLING=fCK_INT, N=2 1001: Sampling frequency fSAMPLING=fDTS/8, N=8 0010: Sampling frequency fSAMPLING=fCK_INT, N=4 1010: Sampling frequency fSAMPLING=fDTS/16, N=5 0011: Sampling frequency fSAMPLING=fCK_INT, N=8 1011: Sampling frequency fSAMPLING=fDTS/16, N=6 0100: Sampling frequency fSAMPLING=fDTS/2, N=6 1100: Sampling frequency fSAMPLING=fDTS/16, N=8 0101: Sampling frequency fSAMPLING=fDTS/2, N=8 1101: Sampling frequency fSAMPLING=fDTS/32, N=5 0110: Sampling frequency fSAMPLING=fDTS/4, N=6 1110: Sampling frequency fSAMPLING=fDTS/32, N=6 0111: Sampling frequency fSAMPLING=fDTS/4, N=8 1111: Sampling frequency fSAMPLING=fDTS/32, N=8
3:2	IC1PSC	RW	0	Input capture 1 prescaler These 2 bits define the prescaler coefficient for the CC1 input (IC1). Once CC1E=0 (in TIMx_CCER register), the prescaler is reset. 00: no prescaler, one capture triggered for every edge detected on the capture input; 01: capture triggered every 2 events; 10: a capture triggered every 4 events; 11: capture triggered every 8 events.
1:0	CC1S	RW	0	Capture/Compare 1 Selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: the CC1 channel is configured as an output; 01: CC1 channel is configured as input, IC1 is mapped on TI1; 10: Reserved; 11: Reserved; Note: CC1S is writable only when the channel is off (CC1E=0 of TIMx_CCER register).

16.3.6. TIMx Capture/Compare Enable Register (TIMx_CCER)

Address offset:0x20

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1NP	Reserved	CC1P	CC1E
Reserved												RW	Reserved	RW	RW

Bit	Name	R/W	Reset Value	Function
15:4	Reserved, always 0			
3	CC1NP	RW	0	Capture/Compare 1 complementary output polarity When the CC1 channel is configured as an output: CC1NP must be held at 0; When CC1 channel is configured as input: CC1NP is used in conjunction with CC1P to act on TIxFP1 polarity (refer to CC1P description)
2	Reserved, always 0			
1	CC1P	RW	0	Capture/Compare 1 output polarity CC1 channel configured as output: 0: OC1 active high; 1: OC1 low active. CC1 channel is configured as input: The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 as trigger or capture signals. 00: Non-inverting/rising edge: TIxFP1 rising edge active (triggered in capture mode); 01: Inverted/falling edge: TIxFP1 falling edge valid (triggered in capture mode); 10: Reserved, do not use this configuration. 11: Non-inverting/dual edge Both TIxFP1 rising and falling edges are valid (triggered in capture mode);
0	CC1E	RW	0	Capture/Compare 1 output enable CC1 channel configured as output: 0: Turn off - OC1 disable output. 1: Enables - OC1 signal output to the corresponding output pin. CC1 channel configured as input: This bit determines whether the counter value can be captured into the TIMx_CCR1 register. 0: Capture disable; 1: Capture enable.

Table 16-1 Control bits for output channel OCx

CCxE bit	OCx output status
0	Output disable (Disconnected from timer) OCx=0. OCx_EN=0

1	OCx=OCxREF+Polarity, OCx_EN=1

Note: The status of the external I/O pins connected to the OCx channel depends on the OCx channel status and the GPIO registers.

16.3.7. TIMx counter (TIMx_CNT)

Address offset:0x24

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	Counter value

16.3.8. TIMx Prescaler (TIMx_PSC)

Address offset:0x28

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	PSC	RW	0	Prescaler value The clock frequency of the counter (CK_CNT) is equal to fCK_PSC/(PSC[15:0]+1). PSC contains the value loaded into the current prescaler register each time an update event is generated; the update event consists of the counter being The update event includes the counter being cleared '0' by the UG bit of TIM_EGR or by a slave controller operating in reset mode.

16.3.9. TIMx Auto Reload Register (TIMx_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	Prescaler value The ARR contains the value that will be loaded into the actual autoreload register. The counter does not work when the autoreload value is empty.

16.3.10. TIMx Capture/Compare Register 1 (TIMx_CCR1)

Address offset:0x34

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1															
RW															

Bit	Name	R/W	Reset Value	Function
15:0	CCR1	RW	0	Capture/Compare 1 value If the CC1 channel is configured as an output: CCR1 contains the value loaded into the current Capture/Compare 1 register (Preload value). If the preload function is not selected in the TIMx_CCMR1 register (OC1PE bit), the written value is immediately transferred to the current register. Otherwise, this preload value is transferred to the current capture/compare 1 register only when an update event occurs. The current capture/compare register participates in the comparison with the counter TIMx_CNT and generates an output signal on the OC1 port. If the CC1 channel is configured as input: CCR1 contains the counter value transferred from the last input capture 1 event (IC1).

16.3.11. TIMx option register (TIMx_OR)

Address offset:0x50

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TI1_RMP	
Reserved														RW	

Bit	Name	R/W	Reset Value	Function
1:0	TI1_RMP	RW	0	Timer Input 1 Remap Set and clear by software.

Bit	Name	R/W	Reset Value	Function
				00: TIM channel 1 connected to GPIO, refer to the datasheet for the multiplexing function. 01: Reserved 10: TIM channel 1 connected to the HSE/128 clock 11: TIM channel 1 is connected to the MCU clock output (MCO). This configuration is determined by the setting of MCO[2:0] in the RCC_CFGR register for the source and MCOPRE[2:0] in the RCC_CFGR1 register for the divider.

16.3.12. TIMx Register Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD		ARPE	Reserved			OPM	URS	UDIS	CEN
	Read/Write																							rw		rw				rw	rw	rw	rw
	Reset Value																							0		0				0	0	0	0
0x0C	TIMx_DIER	Reserved																														CC1IE	UIE
	Read/Write																															rw	rw
	Reset Value																															0	0
0x10	TIMx_SR	Reserved																						CC1OF		Reserved						CC1IF	UIF
	Read/Write																							rc_w0								rc_w0	rc_w0
	Reset Value																							0								0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	TImx_EGR	Reserved																														CC1G	UG
	Read/Write																															w	w
	Reset Value																															0	0
0x18	TImx_CCMR1	Reserved																								Reserved	OC1M			OC1PE	Reserved	CC1S	
																										IC1F				IC1PSC			
	Read/Write																									rw				rw	rw	rw	
	Reset Value																									0	0	0	0	0	0	0	
0x20	TImx_CCER	Reserved																												CC1NP	Reserved	CC1P	CC1E
	Read/Write																													rw		rw	rw
	Reset Value																													0		0	0
0x24	TImx_CNT	Reserved																CNT															
	Read/																	rw															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Write	Reserved																															
	Reset Value																	0															
0x28	TIMx_PSC	Reserved																PSC															
	Read/Write																	rw															
	Reset Value																	0															
0x2C	TIMx_ARR	Reserved																ARR															
	Read/Write																	rw															
	Reset Value																	0xFFFF															
0x34	TIMx_CCR1	Reserved																CCR1															
	Read/Write																	rw/ro															
	Reset Value																	0															
0x50	TIMx_OR	Reserved																														TI1_RMP	
	Read/Write																															rw	

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

17. General-purpose timers (TIM 6 and TIM17)

17.1. Introduction

The basic timers TIM6 and TIM7 each contain a 16-bit auto-load counter driven by their respective programmable prescaler.

TIM6 and TIM7 are completely independent and they do not share any resources.

17.1.1. TIM6 and TIM7 main features

The main functions of the TIM6 and TIM7 timers include:

- 16-bit automatic counter loading
- 16-bit programmable (can be modified in real time) prescaler with a dividing factor of the counter clock frequency of any value between 1 and 65536
- Interrupt/DMA generation on the occurrence of an update event (counter overflow)

17.1.2. Module Block Diagram

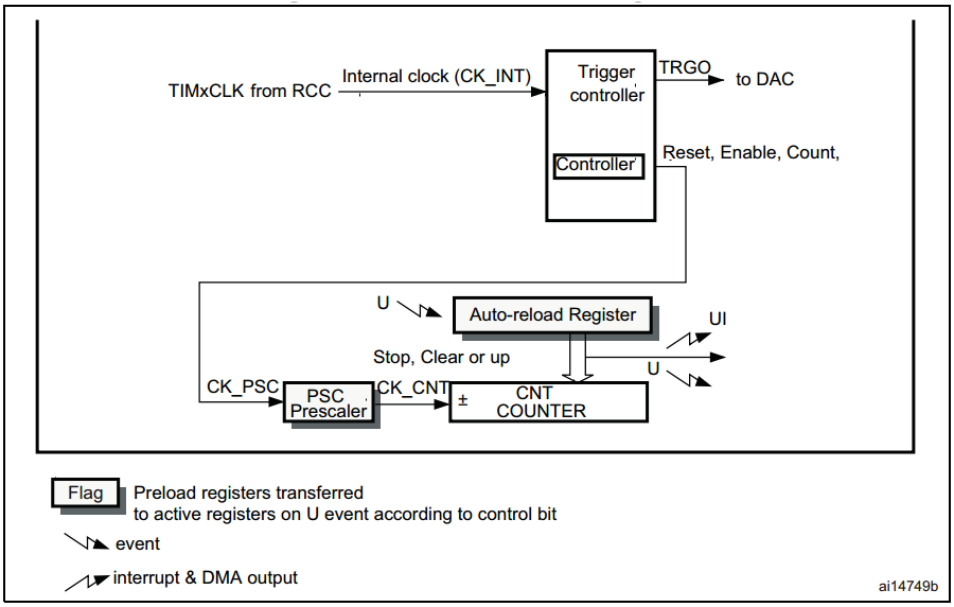


Figure 17-1 TIM6 and TIM7 modules

17.2. TIM6 and TIM7 Functional Descriptions

17.2.1. Time-base unit

The main part of this programmable timer is a 16-bit counter and its associated auto-load register. The clock of the counter can be divided by a prescaler.

The counter, the autoloader register and the prescaler register can be read and written by software, even if the counter is still running.

The time base unit contains:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-load register (TIMx_ARR)

The Autoload register is preloaded and writing or reading the Autoreload register will access its preloaded register. Depending on the setting of the Auto Reload Preload Enable bit (ARPE) in the TIMx_CR1 register, the contents of the preload register are transferred to the shadow register immediately or at each update event (UEV). An update event is generated when the counter reaches an overflow condition and when the UDIS bit in the TIMx_CR1 register equals 0. Update events can also be generated by software. The generation of update events in each configuration will be described in detail later.

The counter is driven by the prescaler divided clock output CK_CNT, which is valid for the counter only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the counter starts counting one clock cycle after the CEN bit in the TIMx_CR1 register is set.

17.2.1.1. Prescaler description

The prescaler can divide the counter clock frequency by any value between 1 and 65536. It is based on a 16-bit counter controlled by a 16-bit register (in the TIMx_PSC register). Because this control register has a buffer, it is able to be changed at runtime. The new prescaling parameter will be applied when the next update event comes.

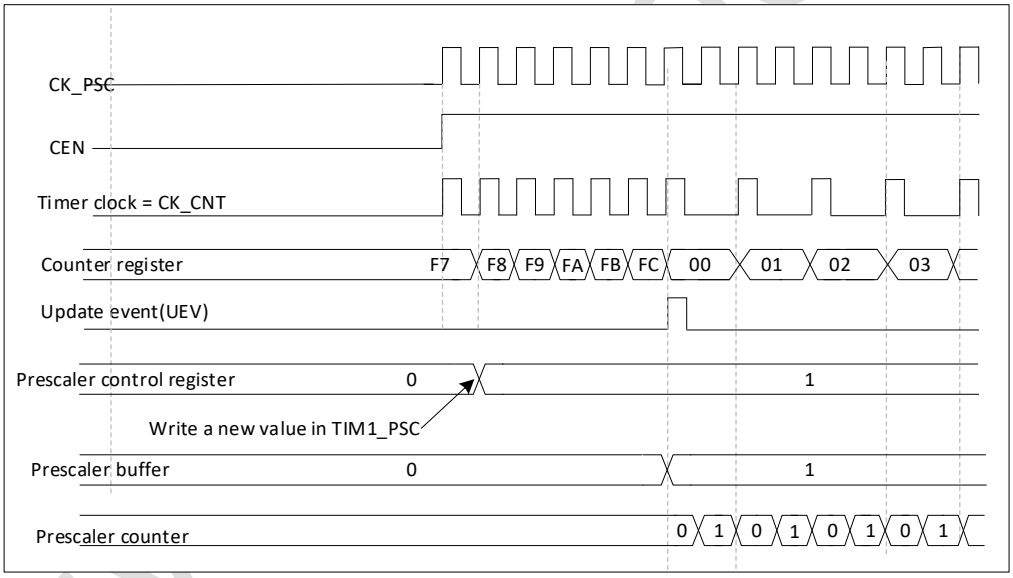


Figure 17-2 Counter timing diagram with prescaler division change from 1 to 2

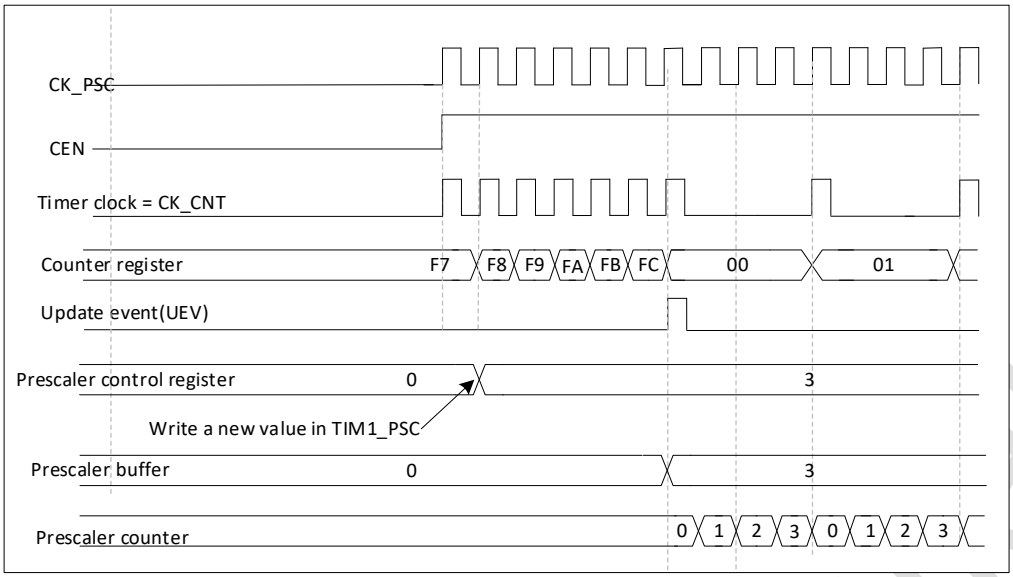


Figure 17-3 Counter timing diagram with prescaler division change from 1 to 4

17.2.2. Counter operation

The counter counts from 0 to the autoload value (the contents of TIMx_ARR), then starts counting from 0 again and generates a count overflow event.

An update event is generated for each count overflow. Setting the UG bit in the TIMx_EGR register (either in software or using the slave mode controller) also generates an update event.

The update event can be disabled by setting the UDIS bit in the TIMx_CR1 register; this prevents the shadow register from being updated when a new value is written to the preload register. An update event will not be generated until the UDIS bit is cleared '0'. Even then, the counter will still be cleared '0' when an update event should be generated, and the internal counter of the prescaler is also cleared '0' (but the value of the prescaler remains unchanged).

In addition, if the URS bit (Select Update Request Source) in the TIMx_CR1 register is set, an update event UEV can be generated by setting the UG bit, but the UIF flag bit will not be set up (i.e., no interrupt or DMA request will be generated).

When an update event occurs, all of the following registers are updated and the hardware sets the update flag bit (UIF bit in the TIMx_SR register) at the same time (based on the URS bit):

- The autoload shadow register is reset to the value of the preload register (TIMx_ARR).
- The prescaler buffer is reset to the value of the preload register (the contents of the TIMx_PSC register).

The following figure gives some examples of how the counter acts at different clock frequencies when TIMx_ARR=0x36.

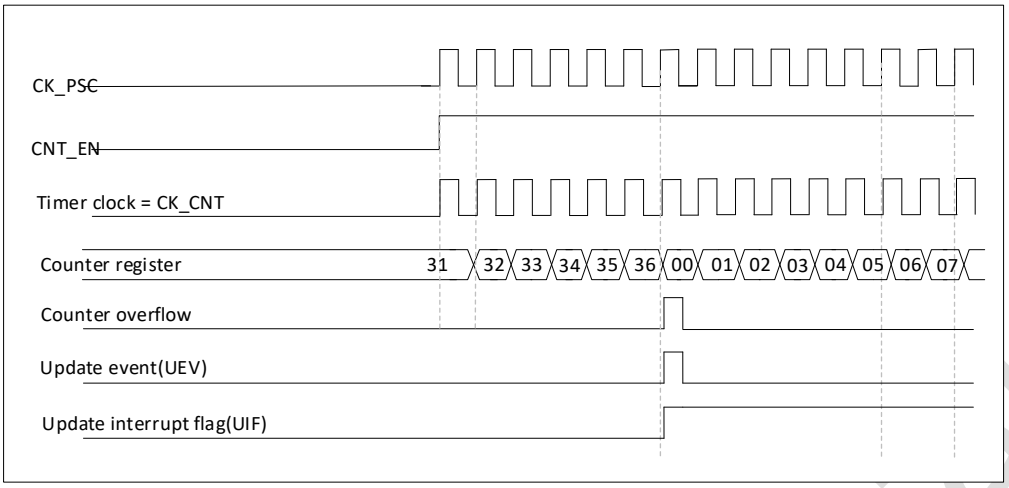


Figure 17-4 Counter timing diagram, internal clock divided by 1

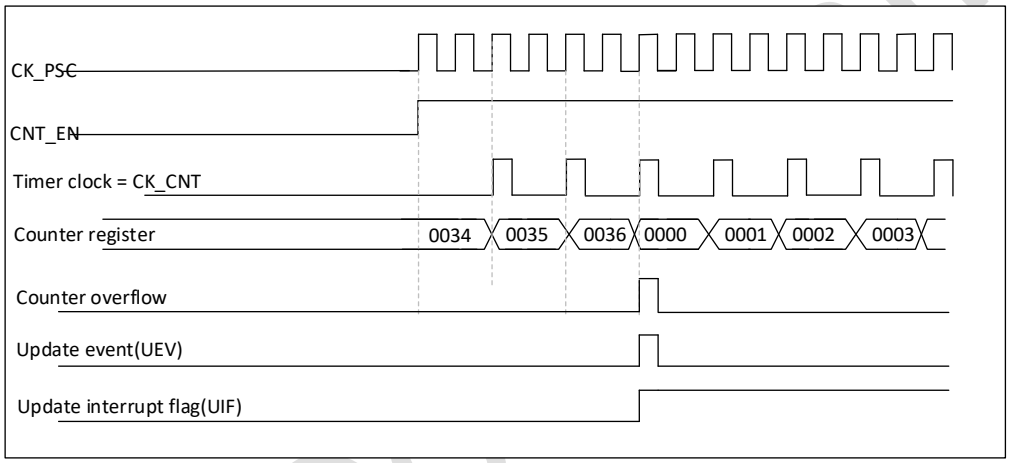


Figure 17-5 Counter timing diagram, internal clock divided by 2

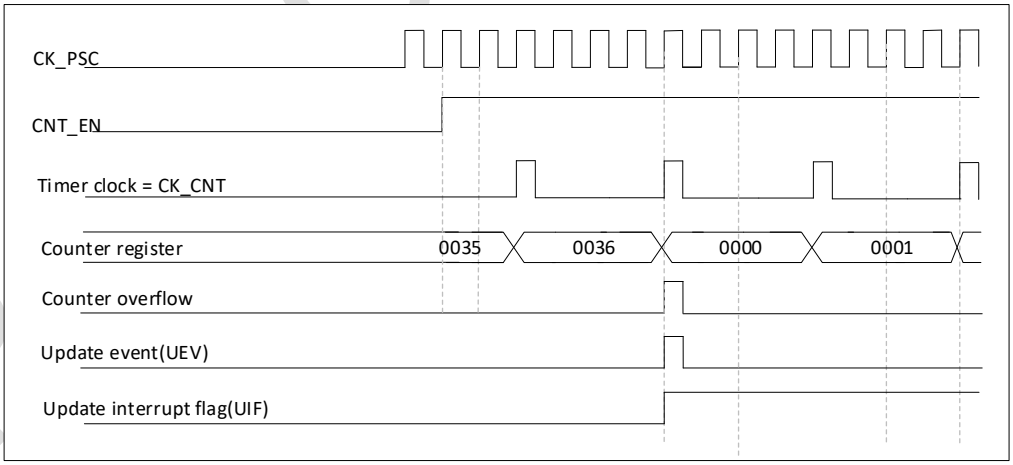


Figure 17-6 Counter timing diagram, internal clock divided by 4

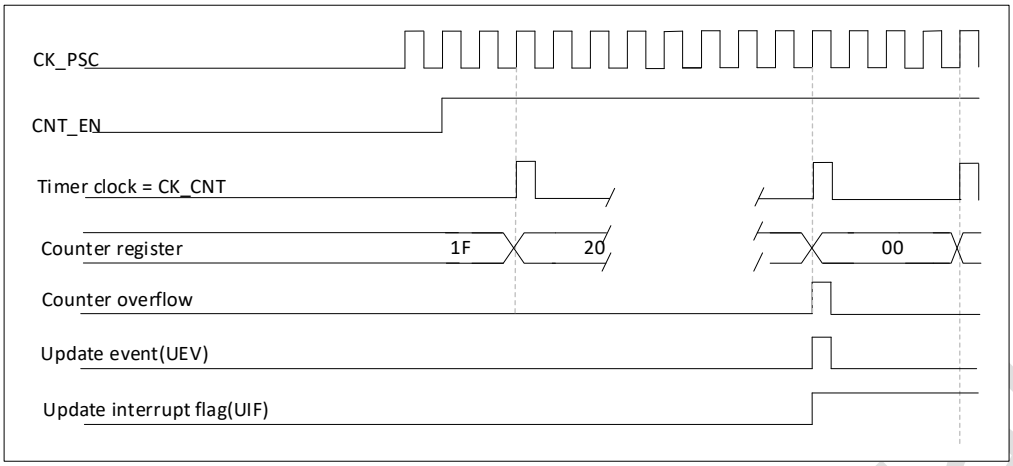


Figure 17-7 Counter timing diagram, internal clock divided by N

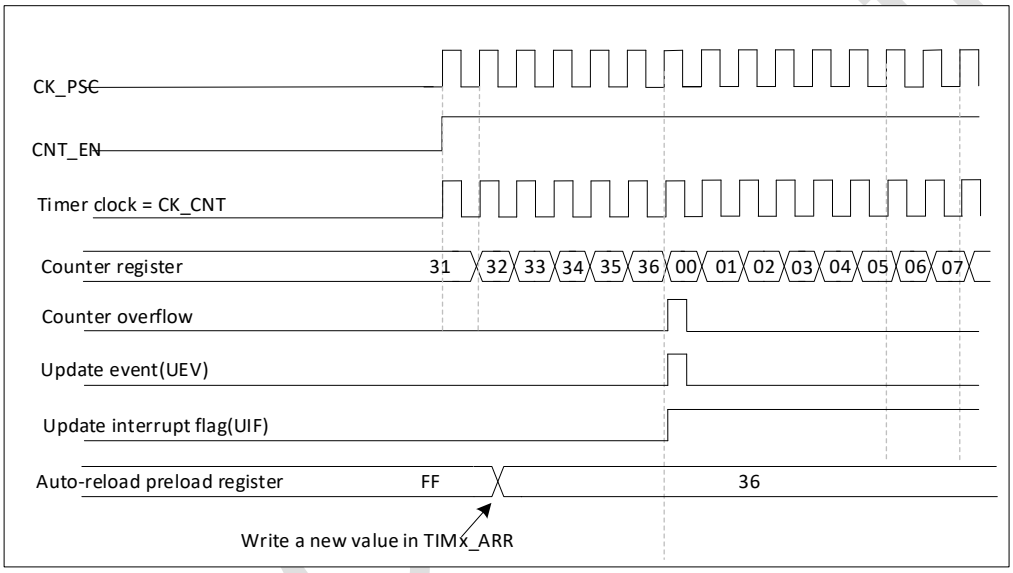


Figure 17-8 Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

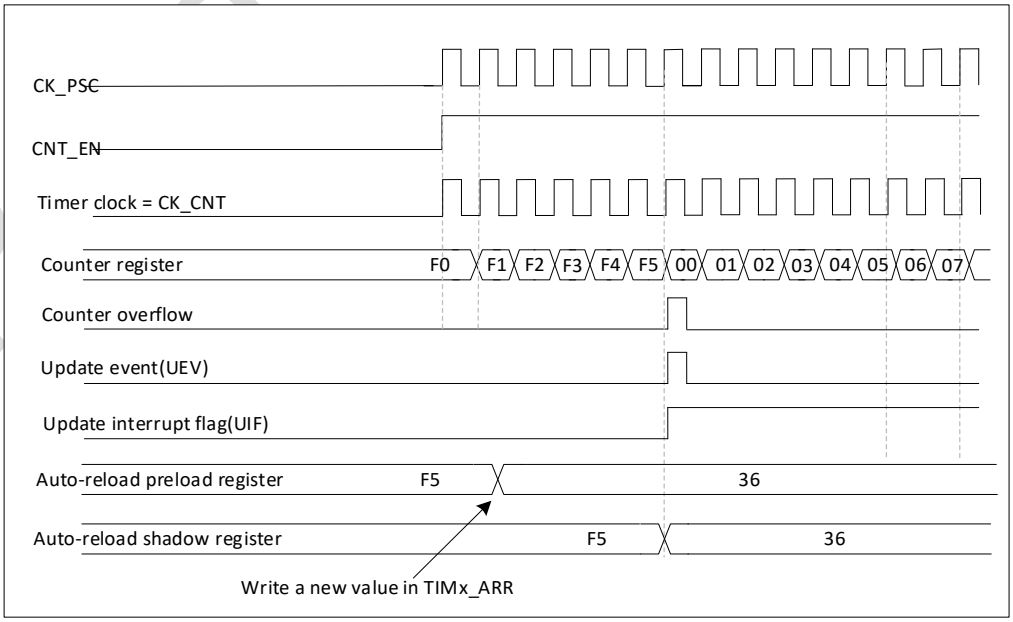


Figure 17-9 Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)

17.2.3. Debug mode

When the microcontroller is in debug mode (Cortex-M4 core stopped), depending on the setting of DBG_TIMx_STOP in the DBG module, the

TIMx counter can either continue its normal operation or stop.

17.3. Register Description

TIM6 Register base address: 0x4000 1000

TIM7 Register base address: 0x4000 1400

17.3.1. TIM6 and TIM7 control register 1 (TIMx_CR1)

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ARPE	Reserved			OPM	URS	UDIS	CEN
Reserved								RW	Reserved			RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	Reserved, always 0			
7	ARPE	RW	0	Auto-reload preload enable bit 0: TIMx_ARR register is not buffered; 1: TIMx_ARR register is buffered.
6:4	Reserved, always 0			
3	OPM	RW	0	One pulse mode 0: the counter does not stop when an update event occurs; 1: The counter stops when the next update event occurs (clearing the CEN bit).
2	URS	RW	0	Update request source The software selects the source of the UEV event with this bit 0: If an update interrupt or DMA request is enabled, the update interrupt or DMA request is generated by any of the following events: - Counter overflow/underflow - Set UG bit - Update generated from the mode controller 1: If an update interrupt or DMA request is enabled, only a counter overflow/underflow generates an update interrupt or DMA request.
1	UDIS	RW	0	Update disable The software allows/disables the generation of UEV events with this bit Update (UEV) events are generated by any of the following events: - Counter overflow/underflow

Bit	Name	R/W	Reset Value	Function
				- Set UG bit - Updates generated from the mode controller Registers with cache are loaded with their preloaded values. (Translation: Update shadow registers) 1: UEV is disabled. no update event is generated and the shadow registers (ARR, PSC, CCRx) maintain their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescaler are reinitialized.
0	CEN	RW	0	Counter enable 0: disable the counter; 1: Enable the counter. Note: The gating mode can only work after the CEN bit is set in software. Trigger mode can set the CEN bit automatically by hardware. In single pulse mode, CEN is automatically cleared when an update event is generated.

17.3.2. TIM6 and TIM7 control register 2 (TIMx_CR2)

Address offset:0x04

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:7	Reserved, always 0			
6:4	MMS	RW	0	Master mode selection These 3 bits are used to select the synchronization information (TRGO) to be sent to the slave timer in master mode. The possible combinations are as follows: 000: Reset - The UG bit of the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by a trigger input (from a mode controller in reset mode), there will be a delay in the signal on TRGO relative to the actual reset. 001: Enable - The counter enable signal CNT_EN is used as the trigger output (TRGO). Sometimes it is necessary to start multiple timers at the same time or to control the enable from a timer over a period of time. The counter enable signal is generated by the logical or of the trigger input signal in CEN control bit and gated mode. When the counter enable signal is controlled by the trigger input, there is a delay on the TRGO

Bit	Name	R/W	Reset Value	Function
				unless Master/Slave mode is selected (see the description of the MSM bit in the TIMx_SMCR register). 010: Update - The update event is selected as the trigger input (TRGO). For example, the clock of a master timer can be used as a prescaler for a slave timer. Note: Slave timer and ADC clocks must be enabled first to receive signals from the master timer and not changed while receiving.
3:0	Reserved, always 0			

17.3.3. TIM6 and TIM7 DMA/interrupt enable registers (TIMx_DIER)

Address offset:0x0C

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:9	Reserved, always 0			
8	UDE	RW	0	Update DMA request enable 0: Disable update DMA requests; 1: Allow the update DMA request.
7:1	Reserved, always 0			
0	UIE	RW	0	Update interrupt enable 0: disable the update interrupt; 1: Allow update interrupt.

17.3.4. TIM6 and TIM7 status registers (TIMx_SR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UIF
Reserved															RC_W0

Bit	Name	R/W	Reset Value	Function
15:1	Reserved, always 0			

Bit	Name	R/W	Reset Value	Function
0	UIF	RC_W0	0	Update interrupt flag This bit is set to '1' by hardware when an update event is generated. It is cleared to '0' by software. 0: no update event is generated; 1: Update interrupt waiting for response. This bit is set to '1' by hardware when the register is updated: - If UDIS=0 of TIMx_CR1 register when the repeat counter value overflows or underflows. - If URS=0 and UDIS=0 of TIMx_CR1 register, the update event is generated when UG=1 of TIMx_EGR register is set, and when the counter CNT is reinitialized by software.

17.3.5. TIM6 and TIM7 event generation registers (TIMx_EGR)

Address offset:0x14

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:1	Reserved, always 0			
0	UG	W	0	Update generation This bit is set to '1' by software and automatically cleared to '0' by hardware. 0: no action; 1: reinitializes the counter and generates an update event. Note that the prescaler counter is also cleared '0' (but the prescaler coefficient remains unchanged).

17.3.6. TIM6 and TIM7 Counters (TIMx_CNT)

Address offset:0x24

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	CNT	RW	0	Counter value

17.3.7. TIM6 and TIM7 Prescaler (TIMx_PSC)

Address offset:0x28

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	PSC	RW	0	Prescaler value The clock frequency of the counter (CK_CNT) is equal to fCK_PSC/(PSC[15:0]+1). PSC contains the value loaded into the current prescaler register each time an update event is generated; the update event consists of the counter being The update event includes the counter being cleared '0' by the UG bit of TIM_EGR or by a slave controller operating in reset mode.

17.3.8. TIM6 and TIM7 Auto Reload Registers (TIMx_ARR)

Address offset:0x2C

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
15:0	ARR	RW	FFFF	Prescaler value The ARR contains the value that will be loaded into the actual autoreload register. The counter does not work when the autoreload value is empty.

17.3.9. TIM6 and TIM7 register maps

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																								ARPE	Reserved			OPM	URS	UDIS	CEN
	Read/Write																									rw				rw	rw	rw	rw
	Reset Value																									0				0	0	0	0
0x04	TIMx_CR2	Reserved																									MMS			Reserved			
	Read/Write																										rw						
	Reset Value																										0						
0x0C	TIMx_DIER	Reserved																							UDE	Reserved							UIE
	Read/Write																								rw								rw
	Reset Value																								0								0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	TIMx_SR	Reserved																															UIF
	Read/Write																																rc_w0
	Reset Value																																0
0x14	TIMx_EGR	Reserved																															UG
	Read/Write																																w
	Reset Value																																0
0x24	TIMx_CNT	Reserved																CNT															
	Read/Write																	rw															
	Reset Value																	0															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	TIMx_PSC	Reserved																PSC															
	Read/Write																	rw															
	Reset Value																	0															
0x2C	TIMx_ARR	Reserved																ARR															
	Read/Write																	rw															
	Reset Value																	0xFFFF															

18. Real time clock (RTC)

18.1. Introduction

The Real Time Clock is a stand-alone timer. The RTC module has a set of counters that count continuously and, with the appropriate software configuration, provide the functionality of a clock calendar. Modifying the value of the counters resets the current time and date of the system.

The RTC module and the clock configuration system (RCC_BDCR register) are in a backup area, i.e., the RTC settings and time are maintained after a system reset or wakeup from standby mode.

After a system reset, access to the backup registers and RTC is disabled, which is to prevent accidental write operations to the backup area (BKP). Performing the following operations will enable access to the backup registers and RTCs:

- Set the PWREN and BKPEN bits of register RCC_APB1ENR to enable the power and backup interface clocks
- Set the DBP bit of register PWR_CR to enable access to the backup registers and the RTC.

18.1.1. Main features

The main functions are as follows:

- Programmable prescaling factor: Scaling factor up to 2^{20}.
- 32-bit programmable counter for longer time periods.
- 2 separate clocks: PCLK1 for APB1 interface and RTC clock
- Three RTC clock sources can be selected as follows:
  - The HSE clock divided by 128;
  - The LSE oscillator clock;
  - LSI oscillator clock
- 2 separate reset types:
  - APB1 interface reset by the system;
  - The RTC core (prescaler, alarm, counter and divider) can only be reset by the backup domain.
- 3 dedicated maskable interrupts:
  - Alarm clock interrupt, used to generate a software programmable alarm clock interrupt.
  - Second interrupt, used to generate a programmable periodic interrupt signal (up to 1 second).
  - Overflow interrupt, to indicate that the internal programmable counter has overflowed and slewed to 0

18.1.2. Module Block Diagram

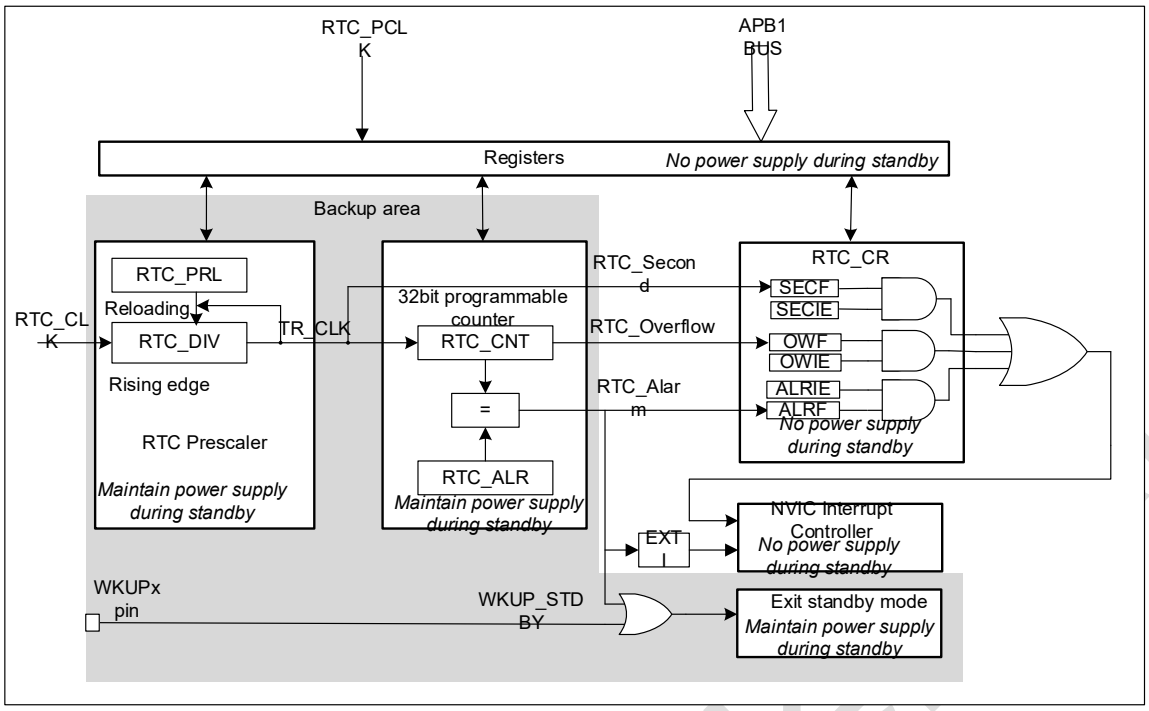


Figure 18-1 RTC module block diagram

The RTC consists of two main parts (see the figure above). The first part (APB1 interface) is used to connect to the APB1 bus. This unit also contains a set of 16-bit registers (RTC_CR, which are actually spread over two addresses) that can be read and written to via the APB1 bus. the APB1 interface is driven by the APB1 bus clock and is used to connect to the APB1 bus.

The other part (the RTC core) consists of a set of programmable counters, divided into two main modules.

The first module is the RTC prescaler module, which is programmable to generate the RTC time reference TR_CLK for a maximum of 1 s. The RTC prescaler module contains a 20-bit programmable divider (RTC prescaler). The RTC generates an interrupt (second interrupt) in each TR_CLK cycle if the corresponding allowable bit is set in the RTC_CR register (second).

The second module is a 32-bit programmable counter that can be initialized to the current system time. The system time is accumulated in TR_CLK cycles and compared to the programmable time stored in the RTC_ALR register. If the corresponding allow bit is set in the RTC_CR control register, an alarm interrupt (alarm) is generated when the comparison matches.

The alarm interrupt is used as a wake-up signal for standby mode and stop mode. The figure above shows the standby mode, after waking up from standby mode. In stop mode, RTC alarm as wakeup requires configuration of EXTI line17 corresponding register.

18.2. Function Description

18.2.1. Register Reset

The RTC_PRL, RTC_ALR, RTC_CNT and RTC_DIV registers can be reset by the backup domain reset signal only. The other system registers (RTC_CR) are reset asynchronously by a system reset or power reset.

18.2.2. Read RTC register

The RTC core is completely independent of the RTC APB1 interface.

Software accesses the RTC prescaler, counter and alarm values through the APB1 interface. However, the associated readable registers are only updated when the signal after synchronization with the rising edge of the RTC clock to the RTC APB1 clock is valid. the same is true for the RTC flags.

This means that if the APB1 interface is ever turned off and the read operation is just after the APB1 is turned back on, the RTC register value read from the APB1 may be corrupted (usually to 0) before the first internal register update. This can happen in the following cases:

- System reset or power reset occurs
- System just woke up from standby mode
- System just woke up from shutdown mode

In all of the above cases, the RTC core remains operational when the APB1 interface is disabled (reset, no clock or power failure).

Therefore, if the APB1 interface of the RTC is ever disabled when the RTC registers are read, the software must first wait for the RSF bit (register synchronization flag) in the RTC_CRL register to be set to '1' by hardware.

Note: The APB1 interface of the RTC is not affected by low power modes such as WFI and WFE.

Description:

1) CPU readable registers include RTC_CR, RTC_CNT and RTC_DIV;

2) The RTC_CR register is the RTC_PCLK domain, which can be read by the CPU at any time to read stable values;

3) RTC_CNT and RTC_DIV are derived from the RTC_CLK domain. RTC_DIV register is updated on the rising edge of each RTC_CLK after RTC operation; RTC_CNT and the flag bits from the RTC_CLK clock domain also use the same update signal as the RTC_DIV register, although this does not change the value of RTC_CNT on each update;

4) RSF is implemented in the RTC_PCLK domain and is set when the pulse signal is valid after synchronization from RTC_CLK to RTC_PCLK;

5) RSF controls only the timing of RTC_CNT and RTC_DIV reads (hardware will not control)

18.2.3. Configure RTC register

The CNF bit in the RTC_CRL register must be set so that the RTC enters the mapping mode before writing to the RTC_PRL, RTC_CNT, and RTC_ALR registers.

In addition, a write operation to any register of the RTC must be performed after the previous write operation. You can determine if the RTC registers are in the process of being updated by querying the RTOFF status bit in the RTC_CR register. Only when the RTOFF status bit is '1', the RTC register can be written.

Configuration process:

- Query the RTOFF bit until the value of RTOFF becomes '1';
- Set the CNF value to 1 to enter configuration mode;
- Perform a write operation to one or more RTC registers;

- Clear the CNF flag bit and exiting configuration mode;
- Query RTOFF until the RTOFF bit changes to '1' to confirm that the write operation has been completed.

Note: The write operation can only be performed when the CNF flag bit is cleared, a process that takes at least 3 RTC_CLK cycles. (The configuration cannot be restarted after 3 RTC_CLK after clearing the CNF flag bit, otherwise a configuration error will occur (controlled by RTOFF=0 at this time))

18.2.4. RTC flag setting

Set the RTC seconds flag (SECF) before changing the RTC counter in every RTC core clock cycle.

Set the RTC overflow flag (OWF) during the last RTC clock cycle before the counter reaches 0x0000.

Set the RTC_Alarm and the RTC alarm flag (ALRF) in the RTC clock cycle before the counter value reaches the alarm register value plus 1 (RTC_ALR+1).

A write operation to the RTC alarm register (RTC_ALR) must be synchronized with the RTC second flag using one of the following procedures:

- Use the RTC alarm clock interrupt and change the RTC alarm clock register (RTC_ALR) and/or the RTC counter register (RTC_CNT) in the interrupt handler.
- Wait for the SECF bit in the RTC control register to be set before changing the RTC alarm register (RTC_ALR) and/or the RTC counter register (RTC_CNT).

18.2.5. RTC Timing

The RTC seconds and alarm clock timings are shown in the following diagram:

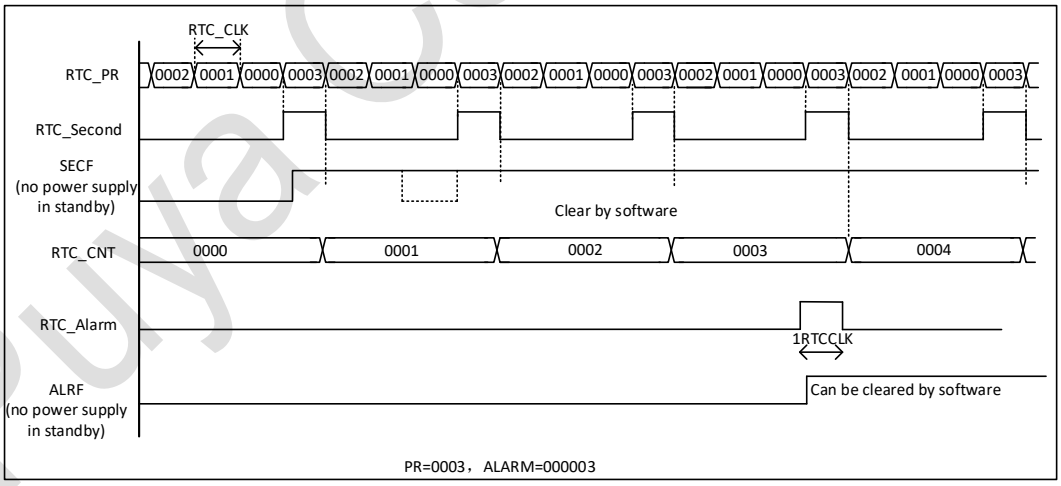


Figure 18-2 RTC Seconds and Alarm Clock Timing

SECF and ALRF are RTC_PCLK domain signals, which are generated by sampling RTC_CLK domain RTC_Sectond and RTC_Alarm signals, respectively.

The RTC overflow timings are shown in the following diagram:

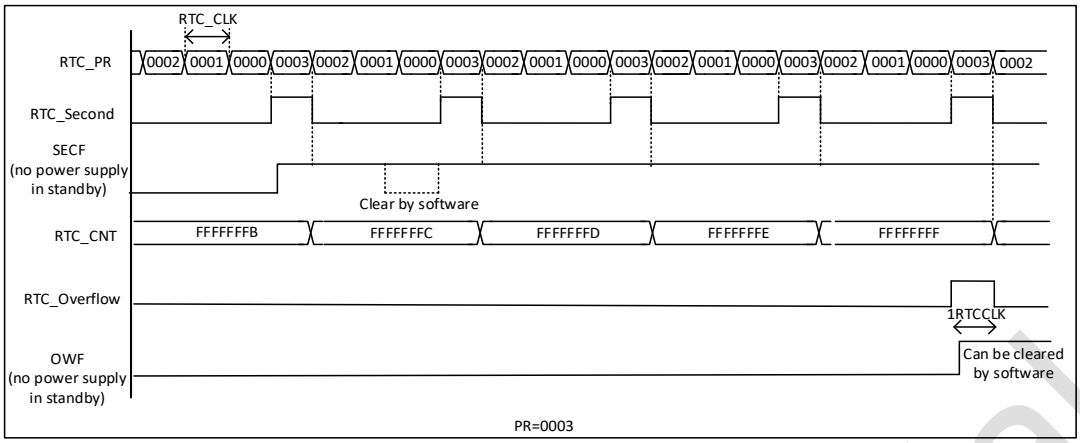


Figure 18-3 RTC Overflow Timing

SECF and OWF are RTC_PCLK domain signals, generated by sampling the RTC_CLK domain RTC_Sectond and RTC_Overflow numbers, respectively.

18.3. Register description (base address 0x4000_2800)

18.3.1. RTC Control Register High (RTC_CRH) (0x00)

Address offset:0x00

Reset value:0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OWIE	ALRIE	SECIE
													RW	RW	RW

This register is reset by system reset.

Bit	Name	R/W	Reset Value	Function
31:3	Reserved			Reserved
2	OWIE	RW	0	Overflow interrupt allowed bit 0: Do not allow overflow interrupts 1: Allow overflow interrupt
1	ALRIE	RW	0	Alarm clock interrupt allowed bit 0: Alarm interrupt is not allowed 1: Allow alarm interrupt
0	SECIE	RW	0	Second interrupt allowed bit 0: No second interrupt allowed 1: second interrupt allowed

These bits are used to mask interrupt requests. Note: After reset, all interrupts are unenabled, so it is possible to write the RTC register after initialization to ensure that there are no interrupt requests that

are pending. However, it is not possible to write the RTC_CRH register while the peripheral is completing a previous write operation (RTOFF=0).

This control register controls the function of the RTC. Certain bits must use a dedicated configuration process to perform write operations.

18.3.2. RTC control register low (RTC_CRL) (0x04)

Address offset:0x04

Reset value:0x0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res.	Res	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RTOFF	CNF	RSF	OWF	ALRF	SECF
										R	RW	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31:6	Reserved	-	-	Reserved
5	RTOFF	R	1	RTC operation OFF, this bit is read-only. The RTC module uses this bit to indicate the status of the last operation performed on its registers (indicating whether the operation is complete or not). If this bit is '0', it indicates that no write operation can be performed on any of the RTC registers. 0: The last write operation to the RTC register is still in progress 1: The last write operation to the RTC register has been completed
4	CNF	RW	0	Configuration flag This bit must be set to '1' by software to enter configuration mode, thus allowing new values to be written to the RTC_CNT, RTC_ALR or RTC_PRL registers. A write operation will only be performed if this bit is set to '1' and cleared '0' by software again. 0: Exit configuration mode (start updating RTC registers) 1: Enter configuration mode
3	RSF	RC_W0	0	Registers synchronized flag. This register is set by hardware and cleared by software. It is set to '1' by hardware when the RTC_CNT register and RTC_DIV register are updated.

Bit	Name	R/W	Reset Value	Function
				This bit must be cleared '0' by software after an APB1 reset, or after the APB1 clock is stopped. To perform any read operation, the user program must wait for this bit to be set '1' by hardware to ensure that RTC_CNT, RTC_ALR or RTC_PRL has been synchronized. 0: register has not been synchronized 1: The register has been synchronized
2	OWF	RC_W0	0	Overflow flag This bit is set to '1' by hardware when the 32-bit programmable counter overflows. If OWIE=1 in the RTC_CRH register, an interrupt is generated. This bit can only be cleared '0' by software, writing '1' is invalid. 0: no overflow; 1: 32-bit programmable counter overflow
1	ALRF	RC_W0	0	Alarm flag This bit is set to '1' by hardware when the 32-bit programmable counter reaches the predefined value set by the RTC_ALR register. If ALRIE = 1 in the RTC_CRH register, an interrupt is generated. This bit can only be cleared '0' by software, writing '1' is invalid. 0: No alarm; 1: With alarm.
0	SECF	RC_W0	0	Second flag When the 32-bit programmable prescaler overflows, this bit is set to '1' by hardware and the RTC counter is added by 1. Therefore, this flag provides a periodic signal (usually 1 second) for the resolution programmable RTC counter. If SECIE = 1 in the RTC_CRH register, an interrupt is generated. This bit can only be cleared by software, writing '1' is not valid. 0: The second flag condition is not valid 1: Second flag condition is established

The function of the RTC is controlled by this control register. When the peripheral is continuing the last write operation (RTOFF=0), it is not possible to write the RTC_CRL register.

Note:

- Any flag bit will remain pending until the appropriate RTC_CR request bit is software reset, indicating that the requested interrupt has been accepted.

- All interrupts are disabled at reset, no pending interrupt requests, and write operations can be performed to the RTC registers (.
- When the APB1 clock is not running, the OWF, ALRF, SECF and RSF bits are not updated (cannot be synchronized).
- The OWF, ALRF, SECF and RSF bits can only be set by hardware and cleared by software.
- If ALRF = 1 and ALRIE = 1, RTC global interrupts are allowed to be generated. If EXTI Line 17 interrupts are allowed to be generated in the EXTI controller, RTC global interrupts and RTC alarm interrupts are allowed to be generated.
- If ALRF=1, the RTC alarm clock interrupt is allowed to be generated if the EXTI Line 17 interrupt mode is set in the EXTI controller; if the EXTI Line 17 event mode is set in the EXTI controller, a pulse is generated on this line (no RTC alarm clock interrupt is generated).

18.3.3. RTC Prescaler Load Register High (RTC_PRLH) (0x08)

The PRL register is used to hold the count value of the RTC prescaler periodicity. This register is write-protected by the RTOFF bit of the RTC_CR register, and only when RTOFF = 1, the CPU is allowed to perform a write operation.

Address offset:0x08

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
31:4	Reserved	-	-	Reserved
3:0	PRL[19:16]	W	0	The RTC prescaler reload value high bits are used to define the clock frequency of the counter according to the following formula: $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$ Note: The value 0 is not recommended, otherwise RTC interrupts and flags cannot be generated correctly

18.3.4. RTC Prescaler Load Register Low (RTC_PRLL) (0x0C)

Address offset:0x0C

Reset value:0x8000

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

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	-	Reserved
15:0	PRL	W	0x8000	The RTC prescaler reload value high bits. These bits are used to define the clock frequency of the counter according to the following formula: $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0] + 1)$ Note: The value 0 is not recommended, otherwise RTC interrupts and flag bits cannot be generated correctly.

18.3.5. RTC prescaler remainder register high (RTC_DIVH) (0x10)

At each TR_CLK cycle, the value of the RTC_PRL register is reloaded into the RTC prescaler counter. The user can read the RTC_DIV register to get the current value of the prescaler counter without stopping the divider counter to get an accurate time measurement.

This register is a read-only attribute and the value of this register will be reloaded by hardware when there is any change in the value of the RTC_PRL or RTC_CNT registers.

Address offset:0x10

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
31:4	Reserved	-	-	Reserved
3:0	RTC_DIV[19:16]	R	0	RTC clock divider remainder high.

18.3.6. RTC prescaler remainder register low (RTC_DIVL) (0x14)

Address offset:0x14

Reset value:0x8000

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

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	-	Reserved
15:0	RTC_DIV[15:0]	R	0x8000	RTC Clock Divider

18.3.7. RTC Count Register High (RTC_CNTH) (0x18)

The RTC module has a 32-bit programmable counter, which is accessed through two 16-bit registers. The count is based on the TR_CLK time base generated by the prescaler as a reference for counting.

The RTC_CNT register is used to store the count value of the counter. The registers are write-protected and the CPU can only perform write operations when RTOFF=1. A write operation to the high 16-bit RTC_CNTH or low 16-bit RTC_CNTL register is loaded directly into the corresponding programmable counter and the RTC prescaler is reloaded. When a read operation occurs, the current value of the counter (system time) is returned.

Address offset:0x18

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	-	Reserved
15:0	RTC_CNT[31:16]	RW	0x0000	RTC core counter high 16bit. When the RTC_CNTH register is read, the high 16 bits of the current value of the RTC counter register is returned, and the register can only be written to by entering configuration mode.

18.3.8. RTC Count Register Low (RTC_CNTL) (0x1C)

Address offset:0x1C

Reset value:0x0000

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

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	-	-	Reserved
15:0	RTC_CNT[15:0]	RW	0x0000	RTC core counter low 16bit. When reading the RTC_CNTL register, it returns the low 16 bits of the current value of the RTC counter register, which can only be written to by entering configuration mode.

18.3.9. RTC alarm register high (RTC_ALRH) (0x20)

When the programmable counter (count) reaches the 32bit value stored in the RTC_ALR register and generates an ALARM interrupt request. This register is write protected by the RTOFF bit and write access is allowed only if RTOFF = 1.

Address offset:0x20

Reset value:0xFFFF

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	ALR[31:16]	RW	0xFFFF	RTC alarm value high 16bit. This register is used to store the high 16 bits of the alarm time written by the software. to write this register you must enter the configuration mode.

18.3.10. RTC alarm register low (RTC_ALRL) (0x24)

Address offset:0x24

Reset value:0xFFFF

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

RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15:0	ALR[15:0]	RW	0xFFFF	RTC alarm value low 16bit. This register is used to store the high 16 bits of the alarm time written by the software. to write this register you must enter the configuration mode.

18.3.11. RTC Register Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RTC_CRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OWIE	ALRIE	SECIE
	Reset value																														0	0	0
0x004	RTC_CRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTOFF	CNF	RSF	OWF	ALRF	SECF
	Reset value																											1	0	0	0	0	0
0x008	RTC_PRLH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRL[19:16]			
	Reset value																													0	0	0	0
0x00C	RTC_PRLL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRL[15:0]															
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x	RTC_	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R																

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10	DIVH	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s																
	Reset value																													0	0	0	0
0x14	RTC_DIVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIV[15:0]															
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	RTC_CNTH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_CNT[31:16]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_CNTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	RTC_ALRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_ALR[31:16]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x24	RTC_ALRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC_ALR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

19. Independent watchdog (IWDG)

19.1. Introduction

Independent Watchdog (IWDG), a module with a high level of security, accurate timing and flexibility of use, can be used to detect and resolve faults caused by software errors and trigger a system reset when the counter reaches a specified timeout value (TIMEOUT).

The independent watchdog (IWDG) is driven by a dedicated low-speed clock (LSI), which remains active even if the master clock fails.

IWDG is best suited for applications that require the watchdog to be used as a watchdog outside of the main program, to be able to work completely independently, and to have low requirements for timing accuracy.

19.2. Main features

- Free-running downward counter
- Clock is provided by a separate RC oscillator (works in STOP and STANDBY modes)
- When the watchdog is activated, a reset is generated when the counter counts to 0x000

19.3. Function Description

19.3.1. Module Block Diagram

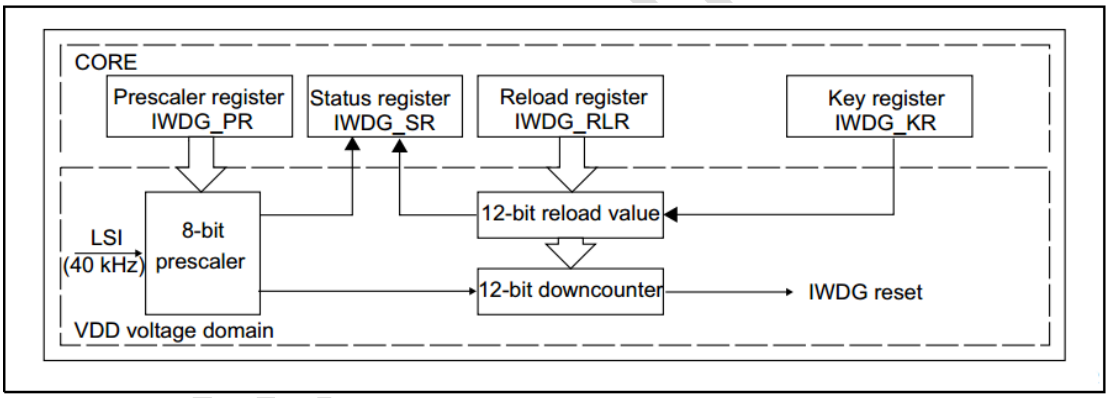


Figure 19-1 IWDG Module Block Diagram

Note: The watchdog function is in the VDD power supply area, i.e. it still works in shutdown and standby mode.

Writing 0xCCCC in the key register (IWDG_KR) starts to enable the independent watchdog; at this point the counter starts to count decreasingly from its reset value 0xFFF. When the counter counts to the end of 0x000, a reset signal (IWDG_RESET) is generated.

Whenever 0xAAAA is written to the key register IWDG_KR, the value in IWDG_RLR is reloaded into the counter to avoid generating an IWDG reset.

Table 19-1

Prescaler	PR[2:0]bit	Minimum time (ms) RL[11:0] = 0x000	Maximum time (ms) RL[11:0] = 0xFFF
/4	0	0.1	409.6

Prescaler	PR[2:0]bit	Minimum time (ms) RL[11:0] = 0x000	Maximum time (ms) RL[11:0] = 0xFFF
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6 或 7)	6.4	26214.4

Note: These times are based on a 40kHz clock. In reality, the internal RC frequency of the MCU will vary between 30kHz and 60kHz. In addition, even if the RC oscillator frequency is accurate, the exact timing still depends on the phase difference between the APB interface clock and the RC oscillator clock, so there will always be a full RC period that is uncertain. A relatively accurate watchdog timeout can be obtained by calibrating the LSI.

19.3.2. Hardware Watchdog

If the power-up loaded option bytes are set to turn on the hardware watchdog, the IWDG power-up is automatically enabled and a reset signal is generated if the IWDG key register is not rewritten by software before the counter counts to its final value.

19.3.3. Register Protection

Write access to registers IWDG Prescaler and IWDG Reload is protected. In order to modify them, the user must first write 0x0000 5555 to the IWDG Key register. writing other numbers to these registers will break the timing, such as writing 0x0000AAAA load, and the registers will be protected again.

If the value of Prescaler register, Reload register is being updated, the status register is reflected.

19.3.4. Debug mode

This function exists only when the system supports DBG_MCU. If the CPU enters debug mode, whether IWDG continues to count normally or enters stop mode depends on the configuration of DBG_IWDG_STOP in the DBG module.

19.3.5. IWDG Register Description

It is possible to operate these peripheral registers in half-word (16-bit) or word (32-bit) mode.

19.3.5.1. Key Value Register (IWDG_KR)

Address offset:0x00

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

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

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

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	RES	-	Reserved
15:0	KEY[15:0]	W	32'h0	Key value. Software must write 0xAAAA to this register at certain intervals, otherwise, the watchdog will generate a reset when the counter counts to 0. 0x5555: indicates that access to IWDG_PR and IWDG_RLR is allowed; 0xCCCC: indicates start IWDG (if hardware watchdog is selected then it is not restricted by this command word).

19.3.5.2. Prescaler register (IWDG_PR)

Address offset:0x04

Reset value:0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31:3	Reserved	RES	-	Reserved
2:0	PR[2:0]	RW	0	Prescale value. Select the prescale value of the counter clock by configuring this register. To change this register, the PVU of the IWDG_SR register must be 0. 000: 4-division frequency; 001: 8-division frequency; 010: 16-division frequency; 011: 32-division frequency; 100: 64-division frequency; 101: 128-division frequency; 110: 256-division frequency; 111: 256-division frequency;

19.3.5.3. Reload register (IWDG_RLR)

Address offset:0x08

Reset value:0x0000 0FFF(reset by Standby mode)

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

Bit	Name	R/W	Reset Value	Function
31:12	Reserved	RES	-	Reserved
11:0	RL[11:0]	RW	0	IWDG counter reload value. When 0xAAAA is written to the IWDG_KR register, the RL value is transferred to the counter. The counter then starts counting decrementally from this value. The watchdog timeout period can be calculated from this RL value and the clock prescaler value. The register can be modified only when IWDG_SR.RVU=0.

19.3.5.4. Status register (IWDG_SR)

Address offset:0x0C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVU	PVU
														R	R

Bit	Name	R/W	Reset Value	Function
31:2	Reserved	RES	-	Reserved
1	RVU	R	0	Watchdog counter reload value update. This bit is set to 1 by hardware to indicate that the reload value is being updated. When the reload value update is complete, this bit is cleared by hardware to zero.
0	PVU	R	0	Watchdog prescaler value update.

Bit	Name	R/W	Reset Value	Function
				This bit is set to 1 by hardware to indicate that the prescaler value is being updated. When the prescaler update is complete, this bit is cleared by hardware.

Note: Before updating IWDG_PR and IWDG_SR.RLR, wait for IWDG_PVU and IWDG_SR.RVU to be 0 respectively. But after updating IWDG_PR and IWDG_RLR, there is no need to wait for IWDG_SR.PVU and IWDG_SR.RVU to be 0, and you can continue to execute the following code.

19.3.5.5. IWDG Register Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[2:0]		
																															0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
																						0	0	0	0	0	0	0	0	0	0	0	0
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVU	PVU
																															0	0	0

20. System window watchdog (WWDG)

20.1. Introduction

Window watchdogs are typically used to monitor, for software faults resulting from deviations from the normal operating sequence of an application caused by external disturbances or unforeseen logic conditions. Unless the value of the decrement counter is flushed before the T6 bit becomes zero, the watchdog circuitry generates an MCU reset when the preset time period is reached. If the 7-bit decrement counter value (in the control register) is flushed before the decrement counter reaches the window register value, then an MCU reset will also be generated. This indicates that the decrement counter needs to be refreshed in a limited time window.

20.1.1. WWDG main features

- Programmable free-running decrement counter
- Conditional Reset
  - A reset is generated when the value of the decrement counter is less than 0x40 (if the watchdog is started).
  - A reset is generated when the decrement counter is reloaded outside the window (if the watchdog is started).
- If the watchdog is enabled and the Early Wakeup Interrupt (EWI) function is allowed to be on, it is generated when the decrement counter equals 0x40, which can be used to reload the counter to avoid a WWDG reset. can be used to reload the counter to avoid a WWDG reset.

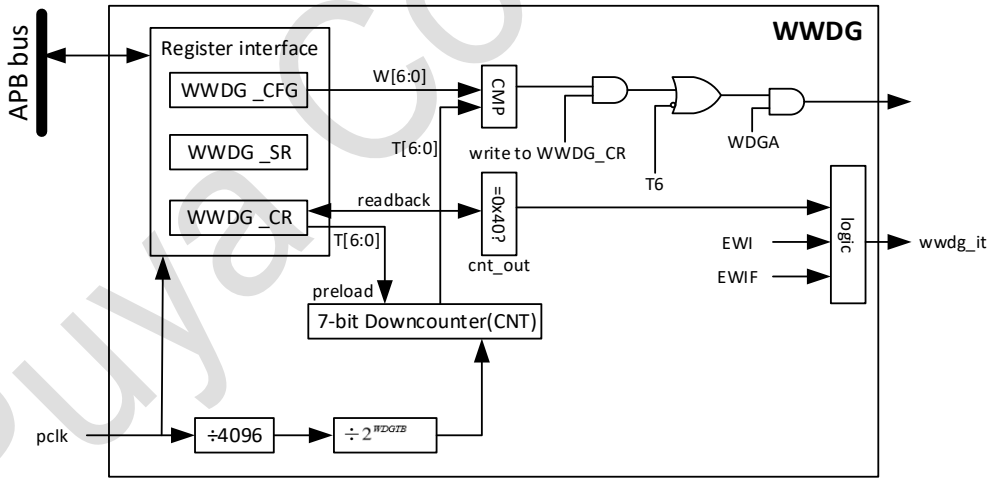


Figure 20-1 Block diagram of WWDG module

20.2. WWDG Function Description

A reset is generated if the watchdog is started (WDGA bits in the WWDG_CR register are set to '1') and when the 7-bit (T[6:0]) decrement counter is flipped from 0x40 to 0x3F (T6 bits are cleared). If the software reloads the counter when the counter value is greater than the value in the window register, a reset is generated.

The application must periodically write to the WWDG_CR register during normal operation to prevent a reset from occurring in the MCU. A write operation can only be performed when the counter value

is less than the value in the window register. The value stored in the WWDG_CR register must be between 0xFF and 0xC0:

- Start the watchdog

After a system reset, the watchdog is always off. Setting the WDGA bit in the WWDG_CR register enables the watchdog, which can then not be turned off again unless a reset occurs.

- Controlling the decrement counter

The decrement counter is in a free-running state and continues to decrement even when the watchdog is disabled. When the watchdog is enabled, the T6 bits must be set to prevent an immediate reset from being generated. the T[5:0] bits contain the number of timings before the watchdog generates a reset; the delay time before a reset varies between a minimum and a maximum value, due to the fact that the prescaler value is unknown when the WWDG_CR register is written. The configuration register (WWDG_CFR) contains the upper limit value of the window: to avoid generating a reset, the decrement counter must be reloaded when its value is less than the value of the window register and greater than 0x3F. Figure 5-1 depicts the working process of the window register.

Another way to reload the counter is to use the Early Wakeup Interrupt (EWI). Setting the WEI bit in the WWDG_CFR register turns on this interrupt. When the decrement counter reaches 0x40, this interrupt is generated and the corresponding interrupt service program (ISR) can be used to load the counter to prevent WWDG reset. Writing '0' to the WWDG_SR register clears this interrupt.

Note: A software reset can be generated using the T6 bit (set WDGA bit to '1' and T6 bit to '0').

20.3. How to write a watchdog timeout program

The window watchdog timeout time can be calculated using the formula provided in Figure 5-1.

Warning: When writing to the WWDG_CR register, always set the T6 bit to '1' to avoid generating an immediate reset.

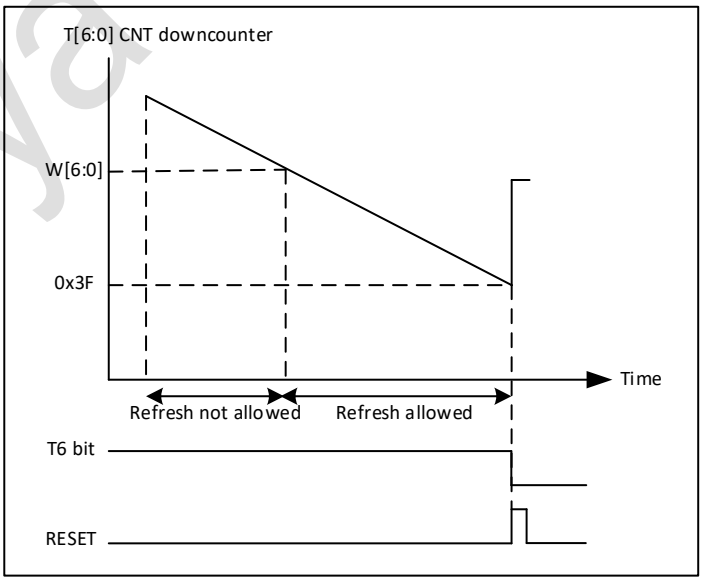


Figure 20-2 Window Watchdog Timing Diagram

The formula for calculating the WWDG timeout value is as follows:

$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WWDGTB[1:0]} \times (T[5:0] + 1)$ (ms)

20.4. Debug mode

When the microcontroller enters debug mode (Cortex-M4 core stopped), the WWDG counter can continue or stop depending on the status of the DBG_WWDG_STOP configuration bit in the debug module.

20.5. Register Description

WWDG register base address: 0x4000 2C00 - 0x4000 2FFF

20.5.1. Control register (WWDG_CR)

Address offset:0x00

Reset value:0x7F

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDGA	T6	T5	T4	T3	T2	T1	T0
Reserved								RS	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	Reserved, always 0			
7	WDGA	RS	0	WDGA: Activation bit This bit is set to '1' by software, but can only be cleared to '0' by hardware after a reset. When WDGA=1, the watchdog can generate a reset. 0: Watchdog disabled 1: Enable watchdog
6:0	T	RW	7F	T[6:0]: 7-bit counter (MSB to LSB) (7-bit counter) These bits are used to store the watchdog counter value. When the counter value changes from 40h to 3Fh (T6 becomes 0) When the counter value changes from 40h to 3Fh (T6 becomes 0), a watchdog reset is generated.

20.5.2. Configuration register (WWDG_CFR)

Address offset:0x04

Reset value:0x7F

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EWI	WDGTB1	WDGTB0	W6	W5	W4	W3	W2	W1	W0
Reserved						RS	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	Reserved, always 0			
9	EWI	RS	0	EWI: Early wakeup interrupt If this bit is set to '1', an interrupt will be generated when the counter value reaches 40h. This interrupt can only be cleared by hardware after reset.
8:7	WDGTB	RW	0	WDGTB[1:0]: Timer base The time base of the prescaler can be set as follows: 00: CK timer clock (PCLK1 divided by 4096) divided by 1 01: CK timer clock (PCLK1 divided by 4096) divided by 2 10: CK timer clock (PCLK1 divided by 4096) divided by 4 11: CK timer clock (PCLK1 divided by 4096) divided by 8
6:0	W	RW	7F	W[6:0]: 7-bit window value These bits contain the window value to be used for comparison with the decrement counter.

20.5.3. Status register (WWDG_SR)

Address offset:0x08

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
Reserved															RC_W0

Bit	Name	R/W	Reset Value	Function
15:1	Reserved, always 0			
0	EWIF	RC_W0	0	EWIF: Early wakeup interrupt flag This bit is set to '1' by hardware when the counter value reaches 40h. It must be cleared by a software write '0'. Writing a '1' to this bit is not valid. If interrupt is not enabled, this bit will also be set to '1'.

20.5.4. WWDG Register Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reset value																									0	1	1	1	1	1	1	1
0x04	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB[1:0]		T[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

21. External Serial Memory Controller (ESMC)

21.1. Introduction

ESMC (External Serial Memory Controller) is a dedicated communication interface for single (Single SPI), dual (Dual SPI), quad (Quad SPI) and octal (Octal SPI) channel SPI interface memories (NOR Flash, PSRAM, etc.). It can operate in either of the following two modes:

- Indirect mode: All operations are performed using ESMC registers (indirect mode)
- Memory mapped mode: External Flash is mapped to the device address space and the system treats it as internal memory (memory mapped mode)

The use of dual memory mode, i.e., simultaneous access to two Qual SPI memories, can achieve twice the throughput and storage capacity similar to the Octal SPI memory.

21.2. Main features

- Two functional modes: indirect and memory mapping
- Can send/receive 8 bits at the same time
  - Dual Flash mode with simultaneous transmit/receive 8 bits by accessing both Flashes in parallel
  - Octal SPI
- SDR and DDR support
- Fully programmable opcodes for indirect and memory mapped modes
- Fully programmable frame formats for indirect and memory-mapped modes
- Integrated FIFO for receive and transmit
- Allows 8-, 16-, and 32-bit data access
- DMA channel for indirect mode operation
- FIFO, interrupt generation on operation completion

21.3. Function Description

21.3.1. Module Block Diagram

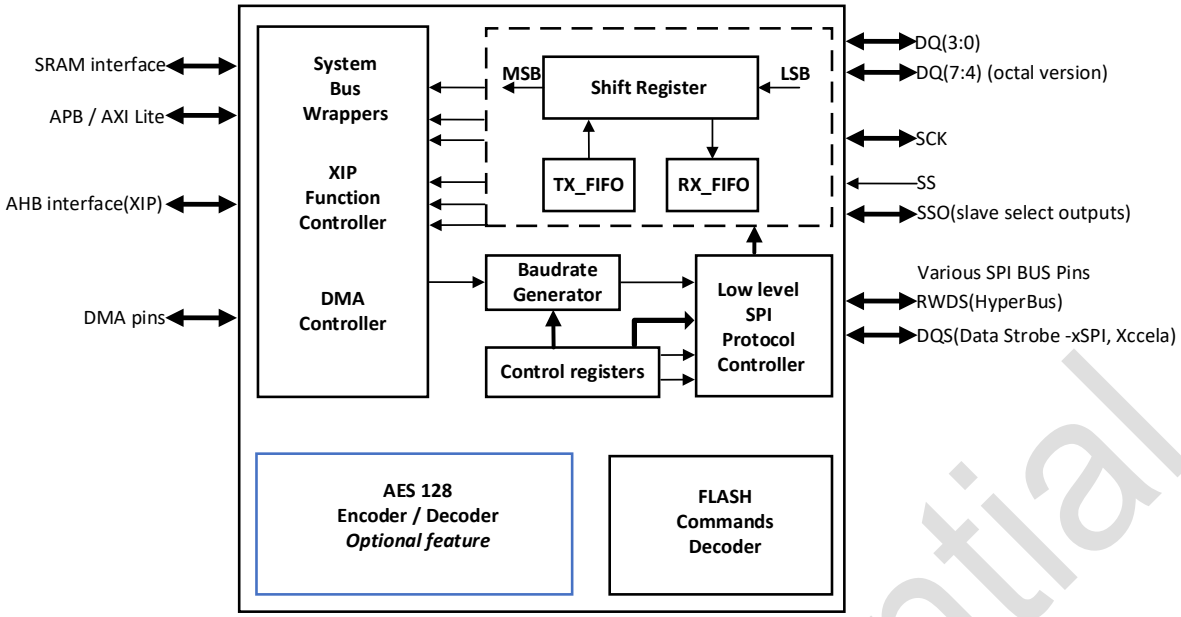


Figure 21-1 ESMC Module Block Diagram

21.3.2. ESMC Functional Description

21.3.2.1. ESMC external memory connection diagram

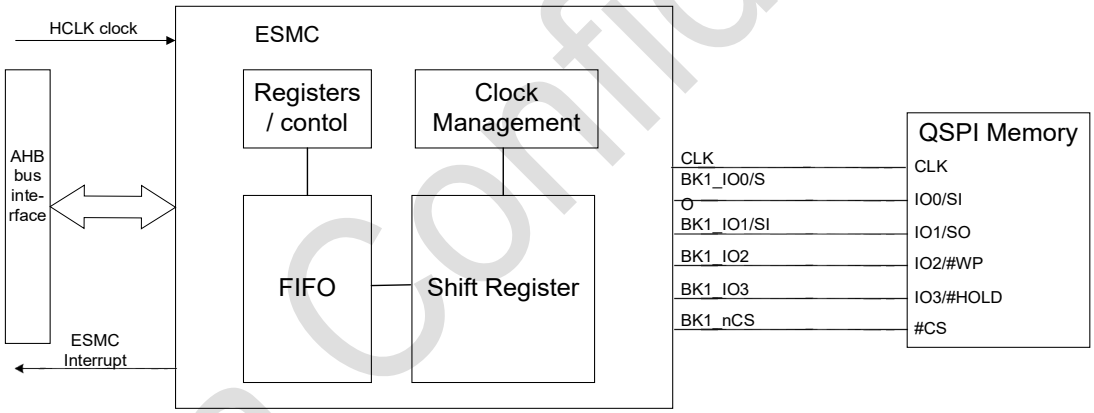


Figure 21-2 ESMC connection diagram (when Dual QSPI Memory mode is disabled)

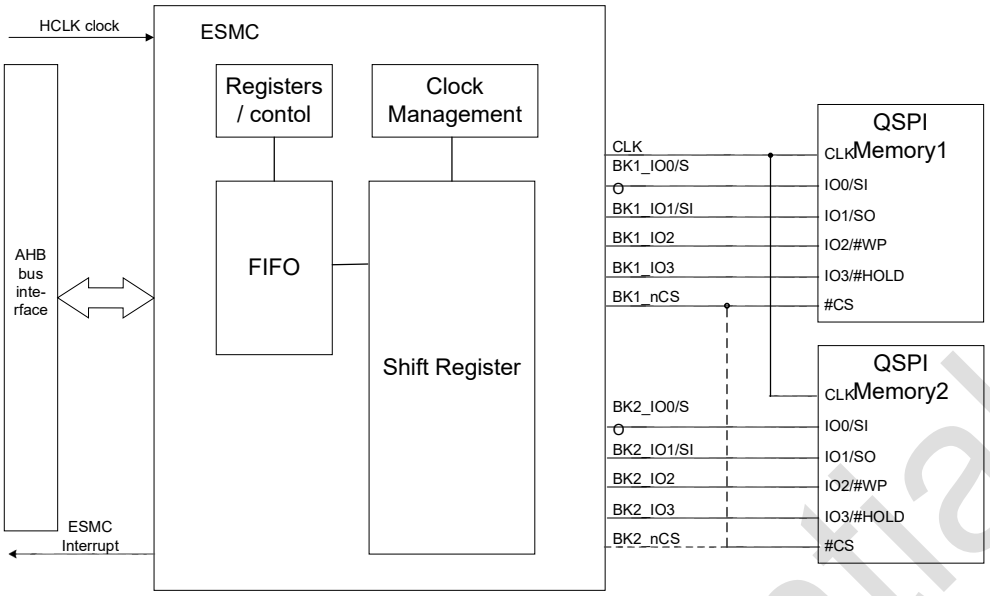


Figure 21-3 ESMC connection diagram (when Dual QSPI Memory mode is set)

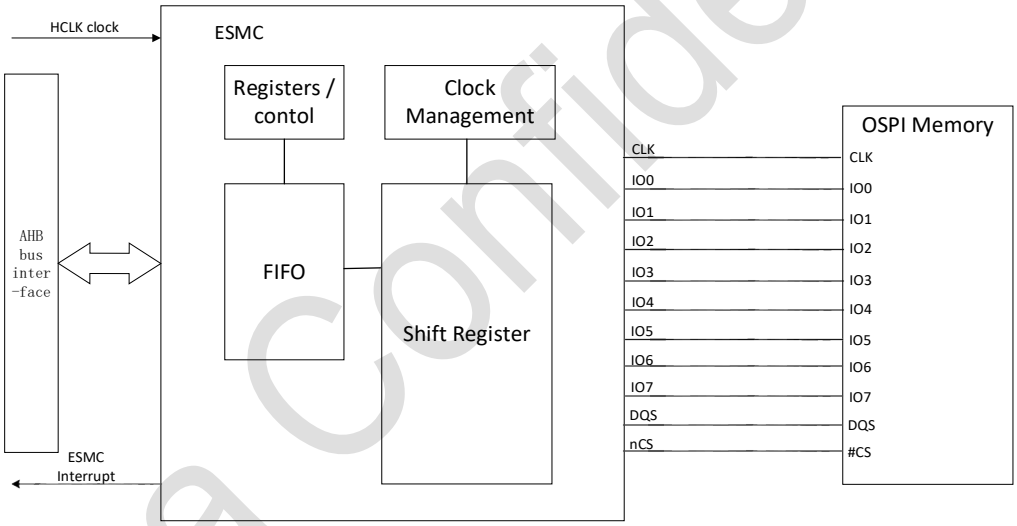


Figure 21-4 ESMC connection diagram (when OSPI Memory mode is set)

21.3.2.2. ESMC Command Sequence

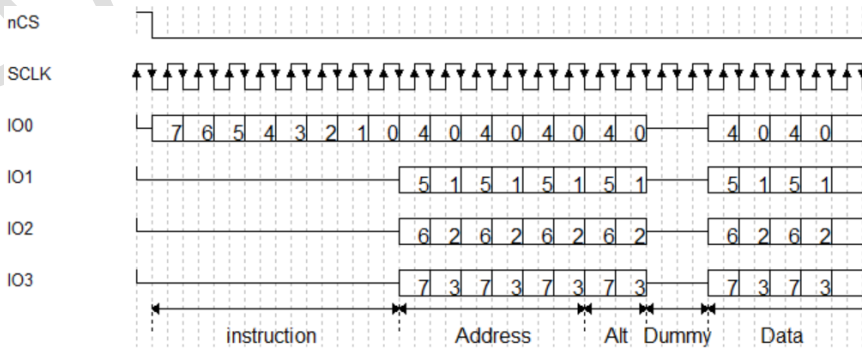


Figure 21-5 QSPI Instruction Example

ESMC uses commands to communicate with external memory. Each command can include five phases: Instruction, Address, Spare Byte, Dummy, and Data. Any of these phases can be configured

to be skipped, but at least one of the instruction, address, spare byte, or data phases must be present.

nCS drops before the start of each command and rises again after the completion of each command.

Instruction phase

In this phase, an 8-bit instruction configured in the instruction field of the ESMC_SFCR[7:0] or ESMC_ADDR24[7:0] (the two INS registers mentioned above are the same register in the circuit) register is sent to the Flash, specifying the type of operation to be performed.

Although most Flash can only receive instructions one bit at a time from the IO0 signal (single SPI mode), the instruction stage can choose to send 2 bits at a time (IO0/IO1 in dual SPI mode), 4 bits at a time (IO0/IO1/IO2/IO3 in quad SPI mode), or 8 bits at a time (IO0/IO1/IO2/IO3/ IO4/IO5/IO6/IO7 in eight SPI mode).

Address Phase

In the address phase, 1 to 4 bytes are sent to the Flash to indicate the address of the operation. The number of bytes of the address to be sent is configured in the 32ADDRBIT, 16ADDRBIT , and 8ADDRBIT bits of the ESMC_CR3 register, and when all the above three bits are 0, the bit width of the address is 24 bits. in the indirect mode, the address bytes to be sent are specified in the ADDR0~2 in the ESMC_ADDR24 and the ESMC_ADDR32 ADDR3 registers in ESMC_ADDR24, while in memory-mapped mode the address is given directly via AHB (from Cortex® or DMA).

The address can be sent 1 bit at a time (via SO in single SPI mode), 2 bits at a time (via IO0/IO1 in dual SPI mode), 4 bits at a time (via IO0/IO1/IO2/IO3 in quad SPI mode) or 8 bits at a time (IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7 in eight SPI mode).

When the NO_ADDR bit of ESMC_DCR register is set, the address phase will be skipped and the command sequence will go directly to the next phase.

Alternate Byte Phase

In the Alternate Byte phase, 1 byte is sent to Flash, which is normally used to control the operation mode. The alternate byte data to be sent is configured in the MREG[7:0] field of the ESMC_ADDR32[15:8] register.

The spare byte phase can be sent 2 bits at a time (via IO0/IO1 in dual SPI mode) or 4 bits at a time (via IO0/IO1/IO2/IO3 in quad SPI mode).

When SENM of ESMC_SOCR or SENM of ESMC_XSOCR = 0, the spare byte phase will be skipped and the command sequence will go directly to the next phase (if available).

Dummy Cycle Phase

During the DUMMY cycle phase, 1-31 cycles are given without sending or receiving any data so that the Flash has time to prepare the data when a higher clock frequency is used. The number of cycles given in this phase is specified in the DUMMY[4:0] field of the ESMC_DCR register. In both SDR and DDR modes, the duration is specified as the number of full CLK cycles.

When DUMMY is zero, the DUMMY cycle phase is skipped and the command sequence goes directly to the data phase (if present).

Data Phase

During the data phase, any number of bytes can be sent to or received from the Flash.

In indirect mode, the number of bytes of data to be sent/received is specified in the ESMC_TCR register.

In indirect write mode, the data sent to Flash must be written to the ESMC_DATA register, while in indirect read mode, the data received from Flash is obtained by reading the ESMC_DATA register.

In memory-mapped mode, the data received from Flash is obtained by reading the ESMC_DATA register.

In DMA mode, the read data is sent back to DMA directly via AHB.

The data phase can send/receive 1 bit (via SO/SI in single SPI mode), 2 bits (via IO0/IO1 in dual SPI mode), 4 bits (via IO0/IO1/IO2/IO3 in quad SPI mode) or 8 bits (IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7 in eight SPI mode) at a time.

21.3.2.3. ESMC Interface Protocol Mode

Single SPI Mode

The traditional SPI mode allows only a single bit to be sent/received serially. In this mode, data is sent to Flash via SO/IO0. Data received from Flash arrives via SI/IO1.

Single SPI mode is used by setting SPIMODE of ESMC_SOCR or ESMC_XSOCR to 000.

Dual SPI Mode

In Dual SPI mode, two bits of data are sent/received simultaneously via IO0/IO1 signals.

Use Dual SPI mode by setting the SPIMODE of ESMC_SOCR or ESMC_XSOCR to 001.

Quad SPI Mode

In Quad SPI mode, four bits of data are sent/received simultaneously via IO0/IO1/IO2/IO3 signals.

Use quad SPI mode by setting the SPIMODE of ESMC_SOCR or ESMC_XSOCR to 010.

Dual Flash Mode

When the 2XQSPI bit (ESMC_CR[2]) is 1, the ESMC is in Dual Flash mode, where two external QUAD SPI Flashes (Flash1 and Flash2) are used to send/receive 8 bits of data per cycle (or 16 bits of data in DDR mode), effectively doubling throughput and capacity.

Each Flash uses the same CLK and optionally the same nCS signals, but each has separate IO0, IO1, IO2, and IO3 signals.

Dual Flash mode can be used in combination with single-bit, double-bit and quad-bit modes as well as SDR or DDR modes.

Each byte 7:4 bits of data is stored in Flash1 and each byte 3:0 bits of data is stored in Flash2.

When reading the Flash status register in dual Flash mode, twice as many bytes should be read compared to reading the same bytes in single Flash mode. This means that if each Flash gives 8 valid bits after the command to fetch the status register, a data length of 2 bytes (16 bits) must be configured for the QUAD SPI and the QUAD SPI receives one byte from each Flash. If the status of each Flash is 16 bits, the QUAD SPI must be configured to read 4 bytes to fetch all status bits of both Flashes in dual Flash mode.

The lowest valid byte of the result (in the data register) is the lowest valid byte of the Flash1 status register, and the next byte is the lowest valid byte of the Flash2 status register. The third byte in the

data register is then the second byte of Flash1, while the fourth byte is the second byte of Flash2 (in the case where the Flash has a 16-bit status register).

In dual Flash mode, an even number of bytes must always be accessed.

Octal SPI Mode

In octal SPI mode, eight bits of data are sent/received simultaneously via IO0/IO1/IO2/IO3/IO4/IO5/IO6/IO7 signals.

Use octal SPI mode by setting the SPIMODE of ESMC_SOCR or ESMC_XSOCR to 100.

SDR Mode

By default, the DDR bit (DDRCMD, DDRADDR, DDRDATA of ESMC_SOCR or ESMC_XSOCR) is 0 and the ESMC operates in single data rate (SDR) mode.

In SDR mode, when the ESMC drives IO0/SO, IO1, IO2, and IO3 signals, these signals only transition with the falling edge of CLK.

When receiving data in SDR mode, the Flash sends data using the falling edge of CLK and the ESMC samples the signals using the rising edge of CLK.

DTR Mode

When the DDR bit (DDRCMD, DDRADDR, DDRDATA of ESMC_SOCR or ESMC_XSOCR) is set to 1, the ESMC operates in double data rate (DDR) mode.

In DDR mode, one bit of data is sent on each falling and rising edge of CLK when ESMC drives IO0, IO1, IO2, IO3, IO4, IO5, IO6, and IO7 signals during the address/alternate byte/data phase.

When receiving data in DDR mode, the Flash sends data using the rising and falling CLK edges. Therefore, after half of the CLK cycle (at the next opposite edge) ESMC samples the signal.

21.3.2.4. Using ESMC

The ESMC operating mode is set by the XIPEN bit (ESMC_CR[6]), XIPEN=0,then it is the indirect mode, otherwise it is the XIP memory mapping mode.

ESMC Indirect Mode

In this mode, the SPI Flash device is read and written through the ESMC control register. Data is read from or written to the SPI Flash from the ESMC data buffer. Therefore, when reading Flash, the CPU reads the same RX FIFO location from ESMC continuously. The following table shows a simplified AMBA bus system with the SPI controller accessible through the AHB memory space. Data from the SPI Flash is accessible through the AHB bus space, and all SPI controller internal registers are accessible through the AHB bus space.

Table 21-1 ESMC Indirect Mode

CPU Addr READ/WRITE	ADDR Bus	SPI Data Bus	Flash Memory
SPI TX/RX Buffer	0x0xx10	[0x0000]	0x0000
	0x0xx10	[0x0001]	0x0001
	0x0xx10	[0x0002]	0x0002
	0x0xx10		
	0x0xx10	[0xFFFF]	0xFFFF

ESMC Memory Mapping Mode

In memory mapping mode, the SPI Flash is mapped to the system bus like any other parallel memory. When the selected memory location is read, the CPU reads the corresponding memory cell from the SPI Flash. the SPI controller uses the SPI Flash command, whose address is the mapped address for AHB access. Using this mode requires the host to configure the operation of the SPI controller, and this setting can also be preset during reset.

Table 21-2 ESMC Memory Mapping Mode

CPU Addr READ/WRITE	ADDR Bus	SPI Data Bus	Flash Memory
SPI Flash location	Addr0	[0x0000]	0x0000
	Addr1	[0x0001]	0x0001
	Addr2	[0x0002]	0x0002
	…		
	Addr N	[0xN]	0xN

In multi-SPI mode, the transfer direction is determined by the read and write commands. Writing and reading data from external SPI devices requires the ESMC to generate SCK clock pulses. When sending data, the host CPU needs to write the data to be sent to the FIFO. When receiving data, the received data is stored in the FIFO. Receiving data requires the host software to set bit 7 (REC:Reception) in the ESMC_DCR register. Before a transfer in any direction can begin, the host needs to address the SPI slave device by the selected SSO/nCS drive being low. To end the current transfer, the SSO/nCS line needs to be driven high.

Indirect mode write data to external serial memory

- Load ESMC control registers (ESMC_CR, ESMC_CR2, ESMC_TCR, ESMC_BAUD, ESMC_SFCR, ESMC_SOCR, ESMC_DCR, ESMC_CR3) with appropriate values
- Load address register with Flash cell address (ESMC_ADDR24, ESMC_ADDR32)
- Load the command register with the Flash command to be sent (ESMC_SFCR[7:0], ESMC_ADDR24[7:0])
- Load to data buffer/FIFO register (ESMC_DATA) using byte data to be sent
- Continue loading the data buffer FIFO until the last byte is sent
- Unselect the memory CS signal by writing to the ESMC control register bit SS_CLEAR_REQUEST (ESMC_CR3[6]) and clearing the SSO

Indirect mode reads data from external serial memory

- Load ESMC control registers (ESMC_CR, ESMC_CR2, ESMC_TCR, ESMC_BAUD, ESMC_SFCR, ESMC_SOCR, ESMC_DCR, ESMC_CR3) with appropriate values
- Load address register with Flash cell address (ESMC_ADDR24, ESMC_ADDR32)
- Load the command register with the Flash command to be sent (ESMC_SFCR[7:0], ESMC_ADDR24[7:0])
- Wait for the data buffer to write data.
- Reads data from the data buffer (ESMC_DATA) until the last byte to be read.
- The SSO is cleared to uncheck the memory CS signal by writing the ESMC control register bit SS_CLEAR_REQUEST (ESMC_CR3[6]).

Memory mapped mode reads data from external serial memory

- Load ESMC control registers (ESMC_CR, ESMC_CR2, ESMC_TCR, ESMC_BAUD, ESMC_XSFCR, ESMC_XSOCR, ESMC_DCR, ESMC_CR3) with appropriate values
- Load address register with Flash cell address (ESMC_ADDR24, ESMC_ADDR32)
- Load the command register with the Flash command to be sent (ESMC_SFCR[7:0], ESMC_ADDR24[7:0])
- Use the AHB interface to access the memory mapped address directly
- Wait for data to be written to the data buffer.
- Read data from the data buffer (ESMC_DATA) until the last byte to be read.
- Uncheck the memory CS signal by clearing the SSO by writing the ESMC control register bit SS_CLEAR_REQUEST (ESMC_CR3[6]).

In the read direction, data transfer is sufficient for memory requirements. Some memories require that Dummy data be transferred to the memory before starting a read. In this case, the user should program ESMC_DCR (DUMMY control register) with the correct number of Dummy cycles to be sent. The IO lines are kept in a high impedance state during the Dummy transfer.

21.3.2.5. ESMC Interrupt

Interrupts can be generated in the following events:

DATA_WAIT

Flash read/write operation is currently being performed and the ESMC state machine is in the wait state

- During the write operation, the command byte and address are sent and the ESMC is waiting to send the next part of the data byte. the FIFO/data buffer is empty.
- In a read operation, ESMC fills the entire data buffer and waits for the host to empty the data buffer.

SPI IDLE

ESMC IDLE Status

FIFO Overflow

FIFO overflow, write FIFO action occurs when FIFO is full

FIFO Full，FIFO Half，FIFO Empty，FIFO Not Empty

FIFO in different states

Independent interrupt enable bits ensure flexibility.

Table 21-3 Interrupt enable bit

Interrupt Event	Event Flag	Enable Control Bit
SPI Busy	DATA_WAIT_FLAG	DATA_WAIT_IE
SPI Idle	IDLE_FLAG	IDLE_IE
FIFO Overflow	FIFO_OVF	FIFO_OVIE
FIFO Full	FIFO_FF	FIFO_FIE
FIFO Half	FIFO_HF	FIFO_HIE
FIFO Empty	FIFO_EF	FIFO_EIE

FIFO Not Empty	FIFO_NEF	FIFO_NEIE

21.3.3. ESMC register description (ONE BYTE access)

ESMC register base address: 0xA000 1000

21.3.3.1. Configuration register (ESMC_CR)

Address offset: 0x00

Reset value: 0xF8

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPIEN	XIP EN	Res	GIE	DMAEN	2xQSPI	Res	SOFTRST
RW	RW		RW	RW	RW		RW

Bit	Name	R/W	Reset Value	Function
7	SPIEN	RW	1	SPI System Enable This bit is set or cleared by software 0: SPI system disable, all internal registers and counters are stopped to reduce power consumption 1: SPI system enable
6	XIPEN	RW	1	Enable/Disable XIP This bit is set or cleared by software. Any change in the value of this bit terminates any SPI operation, clears the TX and RX FIFOs, and drives the ESMC to the idle state. After clearing the XIP EN bit, commands and operations can only be programmed through the register space 0: Disable XIP function and enables indirect mode 1: Enable XIP function, disable indirect mode Note: XIP is the memory mapping mode
5	Reserved			
4	GIE	RW	1	Global interrupt enable 0: Controlled by each interrupt enable bit 1: All interrupts are enabled
3	DMAEN	RW	1	DMA Enable This bit is set or cleared by software to enable DMA requests on TXDMAREQ and RXDMAREQ. When cleared, TXDMAREQ and RXDMAREQ remain zero inactive. Function available only on special request, otherwise this bit is ignored. 0: DMA disable 1: DMA enable
2	2XQSPI	RW	0	Dual Flash Mode (DFM) This bit is set or cleared by software. This bit works in a similar way to OSPI, except that two QSPI flash operations are allowed in this mode. The command and address lines are sent through lines IO0~IO4.

Bit	Name	R/W	Reset Value	Function
				0: Single Flash mode 1: Dual Flash mode
1	Reserved			
0	SOFTRST	RW	0	Software Reset 0: 1: Note: This reset register, in the internal logic, when the bus issues a command to write this register to 1, after an interval of one ESMC module clock, soft_rst is pulled up to 1 at the beginning of the second clock, and when soft_rst is cleared to 0 automatically at the end of the second clock. summary: pull up at an interval of one clock, and automatically cleared to 0 after pulling up and holding one clock.

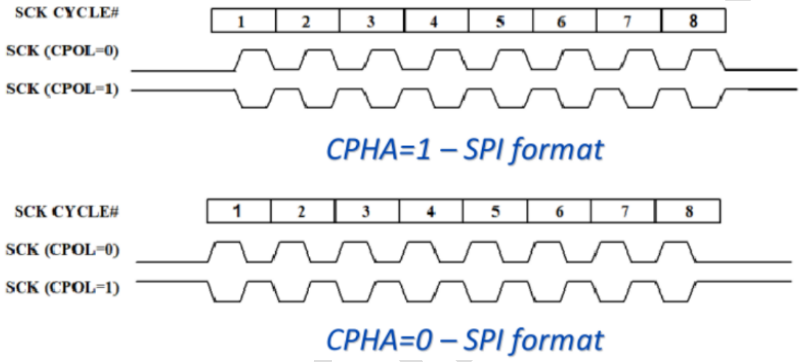


Figure 21-6 Serial interface mode

21.3.3.2. Configuration register 2 (ESMC_CR2)

Address offset: 0x01

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Res						CPOL	CPHA
						RW	RW

Bit	Name	R/W	Reset Value	Function
7:2	Reserved			
1	CPOL	RW	0	Clock polarity, this bit is set or cleared by software 0: When idle, SCK remains low 1: SCK remains high in idle state
0	CPHA	RW	0	Clock phase, this bit is set or cleared by software 0: Sampling data from the second clock edge 1: Sampling data from the first clock edge

21.3.3.3. Transfer Count Register (ESMC_TCR)

Address offset: 0x02

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TCR[7:0]							
RW							

Bit	Name	R/W	Reset Value	Function
7:0	TCR[7:0]	R/W	0x00	Number of data to be read. This register is only used when buffered memory is enabled. In Flash read mode, ESMC can operate in two main read modes: - Read an undefined number of data bytes from Flash-Mode 0 (continuous read) - Read a specified number of data bytes from Flash-Mode 1. By default, the ESMC reads data as long as there is at least one free space in the receiver data buffer. Once the buffer is full, the ESMC enters the wait state and waits for the host to read data from the buffer. The ESMC resumes the read operation as long as there is at least one location in the data buffer that is empty. In Read Specified Byte Count mode, the ESMC reads the number of bytes specified in the TCR register. This mode is only available in non-XIP operation modes. read operation with TCR=0x00 indicates a sequential read mode as described above. In mode 1, after the specified number of bytes are read, the SSO line is deactivated and the ESMC enters the idle state and waits for the next command to be executed. This register allows the user command to receive 1-255 bytes of data. For receiving a larger number of bytes, the user should use continuous receive mode 0.

21.3.3.4. Baud rate register (ESMC_BAUD)

Address offset: 0x03

Reset value: 0x04

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BAUD[7:0]							
RW							

Bit	Name	R/W	Reset Value	Function
7:0	BAUD[7:0]	R/W	0x04	The Baud Rate register is used to define the speed of the SCK clock line relative to the system clock. When loading this register, the user defines the CLK prescaler value used to generate the SCK signal. Acceptable values range from an even number of 2 to 255. The

				values 0x00 and 0x01 are prohibited. Loading 0x00 or 0x01 into the baud rate register will result in generating SCK = CLK/2. The SCK clock frequency generated by the serial SPI transfer is calculated according to the following formula: SCK = CLK/QBAND

21.3.3.5. SPI Flash Command Register (ESMC_SFCR)

Address offset: 0x04

Reset value: 0x0B

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INS0[7:0]							
RW							

Bit	Name	R/W	Reset Value	Function
7:0	INS0[7:0]	RW	0x0B	SPI command (Command Only) The command format contains only the command byte. The address, standby, Dummy, and data bytes are not sent. By default, it is sent in extended SPI format, but it can also be sent in dual mode, quad mode, or QCTAL mode, respectively, for the selected SPI operation mode. The command is never sent in DDR mode, regardless of the DDR control bit value.

21.3.3.6. SPI Output Control Register (ESMC_SOCR)

Address offset: 0x05

Reset value: 0x02

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE1	SPI_MODE0
RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
7	DDRDATA	RW	0	When set, enables dual data rate operation during data reception/transmission. When this bit is set, data is shifted in/out on both SCK clock edges.
6	DDRADDR	RW	0	When set, enables dual data rate operation during Flash address byte transfers. When this bit is set, data is shifted in/out on both SCK clock edges.
5	DDRCMD	RW	0	When set, enables dual data rate operation during Flash command byte transfers. When this bit is set, data is shifted in/out on both SCK clock edges.
4	MULTIADDR	RW	1	When set, fast I/O reading is enabled, where the address bits are sent in double, quad or octal format, respectively, and if MULTICMD is cleared, it means that the address is sent in single SPI mode.

Bit	Name	R/W	Reset Value	Function
				In non-extended SPI transfers (MULTI_COMM=1), the address byte is sent in the same format as the data byte.
3	MULTICMD	RW	1	When not set, extended SPI operation is enabled. This means that Flash commands and addresses are sent in single SPI mode via the DQ0 line, regardless of octal, quad and dual SPI bit values.
2	SENM	RW	0	In indirect mode, MREG[7:0] is sent after ADDR (8-bit, 16-bit, 24-bit or 32-bit).
1:0	SPIMODE[1:0]	RW	0x2	Set the SPI transfer format. 2'b00 SINGLE SPI 2'b01 DUAL SPI 2'b10 QUAD SPI 2'b11 OCTAL SPI

The MUL_COMM bits enable the extended SPI protocol, where commands and addresses are sent on DQ0 lines only and data is sent/received on the programmed lines (IO[1:0] when dual SPI is selected, IO[3:0] when quad is set, and IO[7:0] when octal is set). A dual/quad/octal SPI bit set with multiple communication bit sets means that the dual/quad/octal modes of the SPI are selected separately. The following figure shows the difference in data transfer for a particular mode.

The following figure shows several supported Flash read/write operations and their formats. The programmer needs to manually control the SPI command, address, virtual and data lines as described below. The control registers provided allow any combination of Flash commands to be set, including multi-SPI line commands, DDR options for all commands, addresses and data, and any number of virtual cycles.

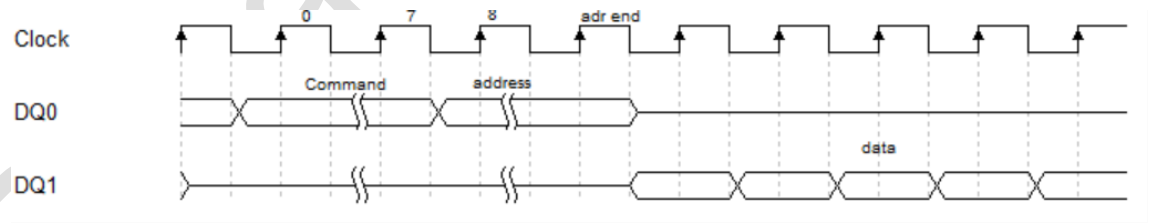


Figure 21-7 Quick read command I/O Fast Read =0; MULTI COMM =0; SPI_MODE=2’b00

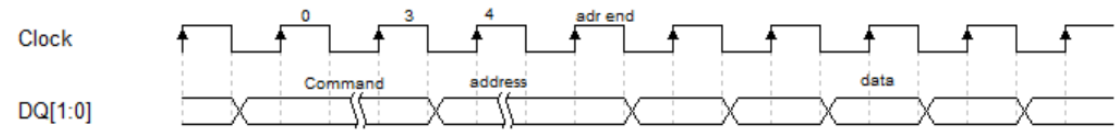


Figure 21-8 Quick read command DUAL I/O Fast Read =0; MULTI COMM =1; SPI_MODE=2’b01

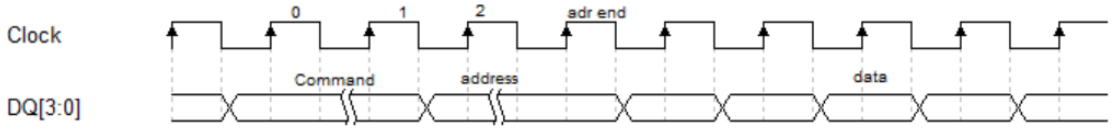


Figure 21-9 Quick read command QUAD I/O Fast Read =0; MULTI COMM =1; SPI_MODE=2'b10

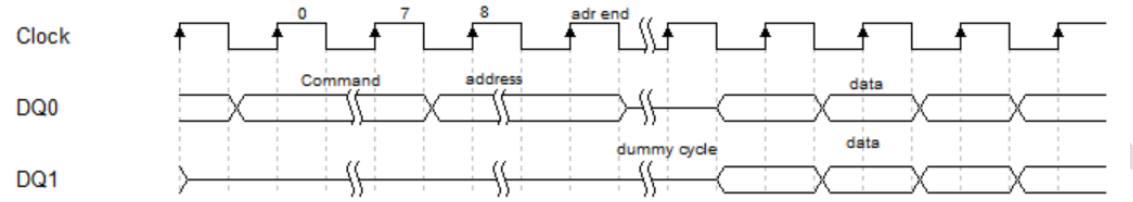


Figure 21-10 Dual output, fast read command I/O Fast Read =0; MULTI COMM =0; SPI_MODE=2'b01; DUMMY !=0

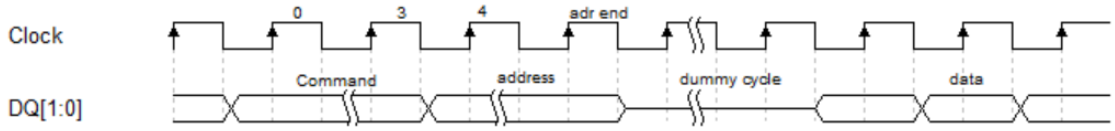


Figure 21-11 Dual output, fast read command I/O Fast Read = 0;MULTI COMM = 1;SPI_MODE = 2'b01; DUMMY != 0

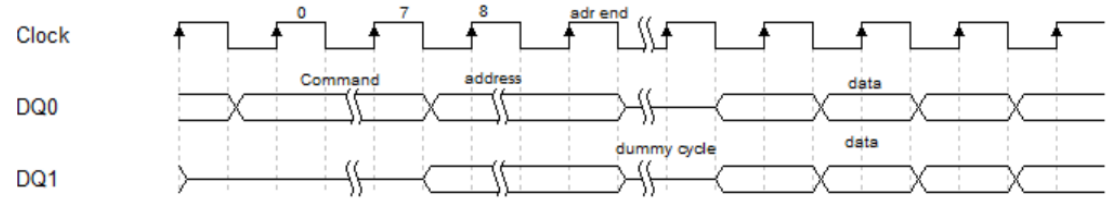


Figure 21-12 Dual I/O, fast read command I/O Fast Read = 1; MULTI COMM = 0; SPI_MODE = 2'b01;DUMMY != 0

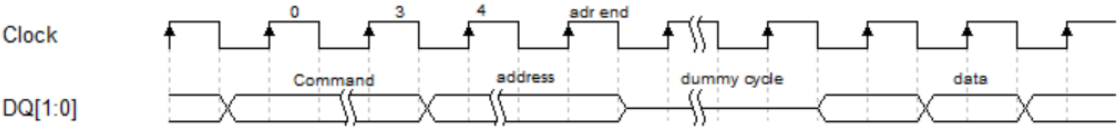


Figure 21-13 Dual I/O, fast read command I/O Fast Read = 1; MULTI COMM = 1; SPI_MODE = 2'b01;DUMMY != 0

Four- and eight-channel read operations are similar to dual-channel, except that data is transferred over multiple I/O lines.

21.3.3.7. Dummy control register (ESMC_DCR)

Address offset: 0x06

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
REC	NO_CMD	NO ADDR	DUMMY[4:0]				
RW	RW	RW	RW				

Bit	Name	R/W	Reset Value	Function
7	REC	RW	0	Recept Data When set, the ESMC shall execute the Flash read command, and after sending the command, address and (if required) DUMMY byte, the ESMC starts to receive the data byte. If this bit is set, the ESMC receives data (read mode), otherwise the ESMC sends data (write mode). Note:TCR[7]=0 turns on the continuous read mode. the ESMC performs the read operation until the FIFO is completely filled and continues the read command after the FIFO is emptied by the software. Then it resumes data reception.
6	NO_CMD	RW	0	This bit is used in manual operation mode for SPI NOR Flash sequential reads (no address decoder). The command will be omitted and the transfer will start with the address.
5	NO_ADDR	RW	0	After setting the NO_ADDR bit, ESMC does not send address bytes, the command bytes are followed by DUMMY CYCLES (if it is eny) and DATA Bytes
4:0	DUMMY[4:0]	RW	5'b00000	Defines the number of DUMMY cycles generated after any address transfer. 11111: 31 dummy cycles 11110: 30 dummy cycles ... 00001: 1 dummy cycle 00000: No dummy cycles

In some configurations, the serial Flash requires that DUMMY cycles be sent after the address byte. esmc_dcr[4:0] defines the number of DUMMY cycles to be sent and allows a setting of 31 DUMMY cycles. DUMMY will be sent automatically when the command requires DUMMY, but the number of DUMMY cycles to be sent must be defined on bits [4:0] if required for a specific operation. By default, the number of DUMMY cycles is set to 8. However, the programmer should also set the number of DUMMY cycles to be sent when there is a change in the clock frequency or when the data preparation time of the Flash is too long. When DUMMY[4:0]=0, no DUMMY cycles will be sent. ESMC_DCR[4:0], unlike 0, causes the ESMC to send the programmed number of DUMMY cycles after the address field. During the sending of DUMMY cycles, the DQ line is set to a high impedance state and no data is stored in the receiver buffer (RX FIFO) and no data is sent out.

21.3.3.8. Configuration register 3 (ESMC_CR3)

Address offset: 0x07

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SS SET	SS CLEAR	FIFO CLR	Res		32BIT_ADDR	16BIT_ADDR	8BIT_ADDR
RW	RW	RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
7	SS_SET_RQ	RW	0	Slave Select Output Set Request, writing a logic 1 to this bit causes the Slave Select Output (SSxO) selected by SS[3:0} to be driven low. This bit will be automatically cleared once the SSxO output is active.
6	SS_CLR_RQ	RW	0	Slave Select Output Clear Request, writing a logic 1 to this bit causes the Slave Select Output (SSxO) to be driven high. This bit is cleared automatically.
5	FIFO CLR	RW	0	Clear all FIFO memory pointers
4:3	Reserved	RW	0	
2	ADDR32BIT	RW	-	Enable 32-bit address, when ESMC_CR3[2:0]=000, the address bit width is 24bit
1	ADDR16BIT	RW	0	Enable 16-bit address
0	ADDR8BIT	RW	0	Enable 8-bit address

21.3.3.9. Status register (ESMC_SR)

Address offset: 0x14

Reset value: 0x10

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPIF	FIFO OV	Res	IDLE STATE	Res	TX in progress	RX in progress	SS_ACTIVE
R	R		R		R	R	R

Bit	Name	R/W	Reset Value	Function
7	SPIF	R	0	SPI interrupt flag 0: No interrupt request pending 1: At least one interrupt request is pending. This flag will remain high until all interrupt requests are cleared.
6	FIFO overflow	R	0	FIFO overflow flag bit
5	Reserved			
4	IDLE	R	1	ESMC is in IDLE status flag bit
3	Reserved			
2	TXBUSY	R	0	Flag bit for controller busy with memory write operations
1	RXBUSY	R	0	ESMC is busy receiving data from Flash flag bit 1: ESMC is reading data from external Flash.

Bit	Name	R/W	Reset Value	Function
				0: RX FIFO Full has not completed any read operation, or has completed the specified number of bytes, or has stopped continuous read operation
0	SSACTIVE	R	0	This bit is 0 if any SS output is valid

21.3.3.10. Interrupt Flag Register (ESMC_IFR)

Address offset: 0x15

Reset value: 0x02

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADDR DONE	FIFO OV	DATA WAIT	IDLE	FIFO Full	FIFO HALF Full	FIFO Eempty	COMMAND DONE
R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
7	ADDRDONEIF	R	0	Address Done Flag Bit set when the address field has just been sent. The bit is automatically cleared by loading a new command into the SFCR register or by writing a 1 to the ADDR Done flag location.
6	FIFOOIF	R	0	FIFO Overflow Flag - FIFO/data buffer overflow. Data write occurs when the buffer is full and too many bytes are loaded in the buffer.
5	DATAWAITIF	R	0	This bit is 1 to indicate that the ESMC is currently performing a Flash read/write operation and the ESMC state machine is in the wait state, which means: The command byte and address were sent during the write operation and the ESMC is waiting to send the next part of the od data byte. The FIFO/data buffer is empty. During a read operation, the ESMC fills the entire data buffer and waits for the host to empty the data buffer. After the data buffer is emptied or a new data portion is loaded, the bit is cleared to allow the ESMC to resume the current Flash read/write operation. 0: Not in data wait phase, current transfer is not finished. 1: Entering data wait phase, current transfer has been terminated
4	IDLEIF	R	0	Idle status flag. 0: Operation is in progress, command, address, Dummy or data transfer is in progress. 1: ESMC is in idle state.

Bit	Name	R/W	Reset Value	Function
3	FIFOFIF	R	0	FIFO full flag - When set FIFO is completely full, there is no space available in FIFO memory. - When clear 0 ，There is at least 1 empty space in the FIFO.
2	FIFOHIF	R	0	FIFO half flag - FIFO memory is half full.
1	FIFOEIF	R	1	FIFO empty flag 0: 1: (Set when Data Buffer and FIFO are empty)
0	CMDIF	R	0	Command done flag 0: Clear when INS loads a new instruction or writes 1 to this bit 1: Set when instruction byte sending is completed. Set when INS loads a new instruction or writes 1 to this bit.

21.3.3.11. Interrupt enable register (ESMC_IER)

Address offset: 0x16

Reset value: 0x00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADDR DONE IE	FIFO OV IE	DATA WAIT IE	IDLE IE	FIFO Full IE	FIFO HALF Full IE	FIFO Empty IE	COMMAND DONE IE
RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
7	ADDRDONEIE	RW	0	Address done interrupt enable 0: Address completion interrupt disable 1: Address completion interrupt enable
6	FIFOOIE	RW	0	FIFO overflow interrupt enable 0: FIFO overflow interrupt disable 1: FIFO overflow interrupt enable
5	DATAWAITIE	RW	0	Busy interrupt enable 0: BUSY interrupt disable 1: BUSY interrupt enable
4	IDLEIE	RW	0	Idle interrupt enable 0: Idle interrupt disable 1: Idle interrupt enable
3	FIFOFIE	RW	0	FIFO full interrupt enable 0: FIFO full interrupt disable

Bit	Name	R/W	Reset Value	Function
				1: FIFO full interrupt enable
2	FIFOHIE	RW	0	FIFO half interrupt enable 0: FIFO half full interrupt disable 1: FIFO half full interrupt enable
1	FIFOEIE	RW	0	FIFO empty interrupt enable 0: FIFO air break disable 1: FIFO air break enable
0	CMDIE	RW	0	Command done interrupt enable 0: Command completion interrupt disable 1: Command completion interrupt enable

21.3.3.12. XIP SPI Flash Command Register (ESMC_XSFCR)

Address offset: 0x1C

Reset value: 0x0B

This register does not support soft_rst reset

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
XINS[7:0]							
RW							

Bit	Name	R/W	Reset Value	Function
7:0	XINS[7:0]	RW	0x0B	Command in XIP mode

The ESMC XIP Instruction Register holds the Flash instruction code sent during an XIP access. The function of this register is similar to the SFCR register, but it is only used in XIP mode. Writing to this register does not initiate command code transfer. It is necessary to use the AHB to access the memory mapped address before the command is automatically issued.

21.3.3.13. XIP SPI Output Control Register (ESMC_XSOCR)

Address offset: 0x1D

Reset value: 0x02

This register does not support soft_rst reset

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE1	SPI_MODE0
RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
7	DDRDATA	RW	0	When set, enables dual data rate operation during data reception/transmission. When this bit is set, data is shifted in/out on both SCK clock edges.
6	DDRADDR	RW	0	When set, enables dual data rate operation during Flash address byte transfers. When this bit is set, data is shifted in/out on both SCK clock edges.
5	DDRCMD	RW	0	When set, enables dual data rate operation during Flash command byte transfers. When this bit is set, data is shifted in/out on both SCK clock edges.
4	MULTIADDR	RW	0	When set, fast I/O reading is enabled, where the address bits are sent in double, quad, or octal format, respectively, and if MUL_COMM is cleared, it means that the address is sent in single SPI mode. In non-extended SPI transfers (MUL_COMM=1), the address byte is sent in the same format as the data byte.
3	MULTICMD	RW	0	When not set, extended SPI operation is enabled. This means that Flash commands are sent in single SPI mode via the DQ0 line, regardless of octal, quad and dual SPI bit values.
2	SEND_M	RW	0	Memory mapped mode sends MREG[7:0] after ADDR (8-bit, 16-bit, 24-bit or 32-bit).
1:0	SPI_MODE[1:0]	RW	0x2	Set SPI transfer format 2'b00 SINGLE SPI 2'b01 DUAL SPI 2'b10 QUAD SPI 2'b11 OCTAL SPI

The XSOCR register is used to control the ESMC operation in XIP mode. This register has the same function as the SOCR register, but its setting is only used during XIP mode Flash read operations.

XDCR[6] is used to enable/disable sending command bytes to the SPI Flash in XIP mode. When bit 6 is set during XIP access, the ESMC sends only the address and programmed DUMMY cycles, followed by the data byte.

21.3.3.14. XIP Dummy Control Register (ESMC_XDCR)

Address offset: 0x1E

Reset value: 0x00

This register does not support soft_rst reset

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
REC	NO_CMD	NO_ADDR	XDUMMY[4:0]				
RW	RW	RW	RW				

Bit	Name	R/W	Reset Value	Function
7	REC	RW	0	Recept Mode Need to be set for xSPI XIP operation to enable data reception.
6	NOCMD	RES	-	This bit is used in manual operation mode for SPI NOR Flash sequential reads (no address decoder). The command will be omitted and the transfer will start with the address.
5	NOADDR	RW	0	After setting the NO_ADDR bit, ESMC does not send address bytes, the command bytes are followed by DUMMY CYCLES (if it is eny) and DATA Bytes
4:0	XDUMMY[4:0]	RW	5'b0000	DUMMY period in XIP mode Defines the number of DUMMY cycles generated after any address transfer in XIP mode. 11111: 31 dummy cycles 11110: 30 dummy cycles ... 00001: 1 dummy cycle 00000: No dummy cycles

21.3.3.15. XIP configuration register 3 (ESNC_XCR3)

Address offset: 0x1F

Reset value: 0x00

This register does not support soft_rst reset

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Res	SSCLRRQ	Res			32BIT_ADDR	16BIT_ADDR	8BIT_ADDR

Bit	Name	R/W	Reset Value	Function
7	Reserved			Reserved
6	SSCLRRQ	RW	0	Slave Select Output Clear Request, writing a logic 1 to this bit causes the slave select output to be driven high. This bit is cleared automatically.
5	Reserved			
4	Reserved			Reserved
3	Reserved			
2	32BITADDR	RW	0	Enable 32-bit addresses
1	16BITADDR	RW	0	Enable 16-bit address
0	8BITADDR	RW	0	Enable 8-bit address

21.3.4. ESMC register description (FOUR BYTE access)

ESMC register base address:0xA000 1000

21.3.4.1. 24-bit address register (ESMC_ADDR24)

Address offset:0x08

Reset value:0x0000000B

This register can only be accessed by word (32 bits)

The ESMC contains four 8-bit address registers, ADDR3, ADDR2, ADDR1, and ADDR0, allowing addressing of Flash supporting 24-bit and 32-bit address fields. by default, the ESMC is configured to operate with 24-bit addresses.

The address registers can be accessed under 2 different address sets. On 32-bit systems, address registers 2-0 can be accessed within a single 32-bit register and loaded with a Flash command to send. This simplifies Flash data read operations with predefined decoder commands. New commands can be initiated by a single write.

The 32-bit wide ADDR register holds the addresses sent from the ESMC during accesses to the external SPI Flash. By default, only the 24 LSB bits of the address register are sent. After setting the 32-bit EN bit (CR3(2)), the 32-bit address is transferred from the ESMC. The address register value is sent in the currently selected SPI mode extended SPI, single SPI, dual SPI, quad SPI, or octal SPI channel. The address register should be loaded before the new Flash command is loaded into the SFCR register or during the same write sequence. In XIP mode, the value of the address register is affected by the address read from the address bus.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR2[7:0]								ADDR1[7:0]							
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
ADDR0[7:0]								INS[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	ADDR2[7:0]	RW	0x00	24-bit address When CR(14) = 0 (default 24-bit address mode), 24-bit address is used The address register value is sent in the currently selected SPI mode extended SPI, single SPI, dual SPI, quad SPI, or eight channel. The address register should be loaded before a new Flash command is loaded into the SFCR register or during the same write sequence.
23:16	ADDR1[7:0]	RW	0x00	
15:8	ADDR0[7:0]	RW	0x00	
7:0	INS1[7:0]	RW	0x0B	SPI Command Address registers 2-0 are accessible in a single 32-bit register and are loaded with the Flash command to send. New commands can be initiated by a single write. By default, the command is sent to Flash at the beginning of each sequence. by default, it is sent in extended SPI format, but it can also be sent in dual mode, quad mode, or QCTAL mode, respectively, for

Bit	Name	R/W	Reset Value	Function
				the selected SPI operation mode. The command is never sent in DDR mode, regardless of the DDR control bit value. esmc can generate and send FLASH commands in any of the selected formats.

21.3.4.2. 32-bit address register (ESMC_ADDR32)

Address offset:0x0C

Reset value:0x0101 0000

This register can only be accessed on a per-word (32-bit) basis

SS0~1 are used to configure which slave select output should be driven during the SPI master transfer. The contents of the registers are automatically assigned to SS1O-SS0O. This register allows easy SPI operation using up to 2 different SPI Flash.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RW				XSS3	XSS2	XSS1	XSS0	RW				SS3	SS2	SS1	SS0
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
MREG[7:0]								ADDR3[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:28	Reserved	RW	0x00	Reserved
27:24	XSSxO[3:0]	RW	0x01	The SSxO pin is selected in XIP mode for host transmission, and the SSxO pin is also the CS (chip sel) input from the slave.
23:20	Reserved	RW	0x00	Reserved
19:16	SSxO[3:0]	RW	0x01	Indirect mode selects the SSxO pin for host transmission, and the SSxO pin is also the CS (chip sel) input of the slave.
15:8	MREG	RW	0x00	The content of this register is sent after the address byte only if the SEND_M bit is set in the CR3 register.
7:0	ADDR3[7:0]	RW	0x00	32-bit address When CR3(2) = 1 (32-bit address enable), 32-bit address is used The address register value is sent in the currently selected SPI mode extended SPI, single SPI, dual SPI, quad SPI, or eight channel. The address register should be loaded before the new Flash command is loaded into the SFCR register or during the same write sequence.

21.3.4.3. Data register (ESMC_DATA)

Address offset:0x10

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3 [7:0]								DATA2 [7:0]							
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
DATA1 [7:0]								DATA0 [7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	DATA3 [7:0]	RW	0x00	Data sent to/received from external SPI devices Data buffer for data transfer during Flash write operations, and data buffer for data byte storage during read commands. These addresses can be used in both directions. When data buffer memory is enabled, these registers point to the top of the FIFO in both the read and write directions.
23:16	DATA2[7:0]	RW	0x00	
15:8	DATA1[7:0]	RW	0x00	
7:0	DATA0[7:0]	RW	0x00	

21.3.5. ESMC Register map

ESMC base address: 0xA000 1000

When the communication mode is set to HyperBus™, the 0x04, 0x08 address register function is switched to CA0, CA1, CA2 function.

FOUR BYTE access register

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA0001000	ESMC_CR	BAUD[7:0]								TCR[7:0]								Reserved		WREN	SLAVEN	XSPI[1:0]		CPOL	CPHA	SPIEN	XIPEN	DECEN	GIE	DMAEN	2XQSPI	Reserved	SOFTRST
	Read/Write	RW								RW										RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
	Reset Value	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0x04	ESMC_CTL	SSSETRQ	SSCLRRQ	FIFOCLR	Reserved		ADDR32BIT	ADDR16BIT	ADDR8BIT	REC	NOCMD	NOADDR	DUMMY[4:0]					DDRDATA	DDRADDR	DDRCMD	MULTIADDR	MULTICMD	SENDM	SPIMODE[1:0]		INS[7:0]								
	Read/Write	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW	RW	RW	RW	RW	RW							
	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1

	Value																														
0x08	ESMC_ADDR24	ADDR2[7:0]								ADDR1[7:0]								ADDR0[7:0]								INS[7:0]						
	Read/Write	RW								RW								RW								RW						
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
0x0C	ESMC_ADDR32	Reserved				XSS3	XSS2	XSS1	XSS0	Reserved				SS3	SS2	SS1	SS0	MREG[7:0]								ADDR3[7:0]						
	Read/Write					RW	RW	RW	RW					RW	RW	RW	RW	RW								RW						
	Reset Value	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	ESMC_DATA	DATA3 [7:0]								DATA2 [7:0]								DATA1 [7:0]								DATA0 [7:0]						
	Read/Write	RW								RW								RW								RW						
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	ESMC_SR	Reserved								ADDRDONEIE	FIFOOIE	DATAWAITIE	IDLEIE	FIFOFIF	FIFOHIF	FIFOEIE	CMDIE	ADDRDONEIF	FIFOOIF	DATAWAITIF	IDLEIF	FIFOFIF	FIFOHIF	FIFOEIF	CMDIF	SPIF	FIFOv	Reserved	IDLE	Reserved	TXBUSY	RXBUSY	SSACTIVE
	Read/Write									RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0

0x1C	ESMC_XIP	Reserved	SSCLRRQ	Reserved			32BITADDR	16BITADDR	8BITADDR	REC	NOCMD	NOADDR	XDUMMY[4:0]					DDRDATA	DDRADDR	DDRCMD	MULTIADDR	MULTICMD	SENDM	SPIMODE[1:0]		XINS[7:0]							
	Read/Write		RW				RW	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW	RW	RW	RW		RW							
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1

ONE BYTE Access Register

Offset	Register	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
0x00	**CR**	SPIEN	XIP EN		GIE	DMA EN	2xQUAD	-	SOFT RST	0xF8
	R/W	RW	RW		RW	RW	RW		RW	
	Reset Value	1	1		1	1	0		0	
0x01	**CR2**	-	-					CPOL	CPHA	0x00
	R/W							RW	RW	
	Reset Value							0	0	
0x02	**TCR**	TCR[7:0]								0x00
	R/W	RW								
	Reset Value	0								
0x03	**BAUD**	BAUD[7:0]								0x04
	R/W	RW								
	Reset Value	0x04								
0x04	**SFCR0 COMMANDONLY**	INS[7:0]								0x0B
	R/W	RW								
	Reset Value	0x0B								
0x05	**SOCR**	DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE[1:0]		0x02
	R/W	RW	RW	RW	RW	RW	RW	RW		
	Reset Value	0	0	0	0	0	0	0x02		

Offset	Register	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
0x06	**DCR**	REC	NO_CMD	NO ADDR	DUMMY[4:0]					0x00
	R/W	RW	RW	RW	RW					
	Reset Value	0	0	0	0					
0x07	**CR3**	SS SET	SS CLEAR	FIFO CLR			32BIT_ADDR	16BIT_ADDR	8BIT_ADDR	0x00
	R/W	RW	RW	RW			RW	RW	RW	
	Reset Value	0	0	0			0	0	0	
0x14	**SR**	SPIF	FIFO OV		IDLE STATE		TX in progress	RX in progress	SS_ACTIVE	0X10
	R/W	RW	RW		RW		RW	RW	RW	
	Reset Value	0	0		1		0	0	0	
0x15	**IFR**	ADDR DONE	FIFO OV	DATA WAIT	IDLE	FIFO Full	FIFO HALF Full	FIFO Eempty	COMMAND DONE	0x02
	R/W	R	R	R	R	R	R	R	R	
	Reset Value	0	0	0	0	0	0	1	0	
0x16	**IER**	ADDR DONE IE	FIFO OV IE	DATA WAIT IE	IDLE IE	FIFO Full IE	FIFO HALF Full IE	FIFO EmptyIE	COMMAND DONE IE	0x00
	R/W	RW	RW	RW	RW	RW	RW	RW	RW	
	Reset Value	0	0	0	0	0	0	0	0	
0x1C	**XSFCR**	XINS[7:0]								0x0B
	R/W	RW								
	Reset Value	0x0B								
0x1D	**XSOCR**	DDR DATA	DDR ADDR	DDR COMM	MULTI ADDR	MULTI COMM	SEND_M	SPI_MODE[1:0]		0x02
	R/W	RW	RW	RW	RW	RW	RW	RW		

Offset	Register	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset
	Reset Value	0	0	0	0	0	0	0x02		
0x1E	**XDCR**	REC	NO_CMD	NO_ADDR	DUMMY[4:0]					0x00
	R/W	RW	RW	RW	RW					
	Reset Value	0	0	0	0					
0x1F	**XCR3**	-	SSCLRRQ	-	-	-	32BIT_ADDR	16BIT_ADDR	8BIT_ADDR	0x00
	R/W		RW				RW	RW	RW	
	Reset Value		0				0	0	0	

22. SDIO Interface (SDIO)

22.1. Introduction

22.1.1. Main features

The SD/SDIO MMC card host module (SDIO) provides an operational interface between the AHB peripheral bus and multimedia cards (MMC), SD memory cards, SDIO cards, and CE-ATA devices.

The Multimedia Card System Specification is published by the MMCA Technical Committee and is available on the Multimedia Card Association's Web site.

The CE-ATA system specification is available on the CE-ATA Working Group website.

The following features are supported:

- Support SD card version 2.0
- Support SD I/O card version 2.0
- Support MMC version 4.2
- Support CE-ATA version 1.1
- Support command completion signal and interrupt to host processor
- Command completion signal shutdown function

Notes:

1) SDIO does not support SPI mode communication mode
2) The I/O portion of SD cards or composite cards that only support I/O mode cannot support many commands needed in SD memory devices. Some commands here do not work in SD I/O devices, such as the erase command, so SDIO does not support these commands. In addition, some commands in SD memory card and SD I/O card are different, and SDIO does not support these commands.
3) MMC4.1 does not support DDR boot

22.1.2. Module Block Diagram

22.2. Function Description

SDIO contains 2 parts:

- SDIO adapter module: implements all MMC/SD/SD I/O card related functions, such as clock generation, command and data transfer.
- AHB bus interface: Operates the registers in the SDIO adapter module and generates interrupt and DMA request signals.

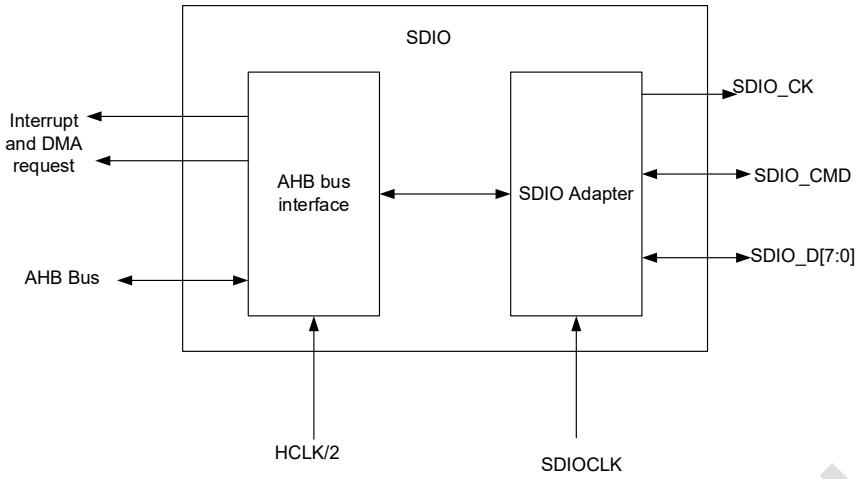


Figure 22-1 SDIO Block Diagram

By default SDIO_D0 is used for data transfer after reset. After initialization the host can change the width of the data bus. If a multimedia card is connected to the bus, SDIO_D0, SDIO_D[3:0] or SDIO_D[7:0] can be used for data transfer. MMC version V3.31 and previous versions of the protocol only support 1-bit data lines, so only SDIO_D0 can be used.

If an SD or SD I/O card is connected to the bus, SDIO_D0 or SDIO_D[3:0] can be used for data transfer via host configuration. All data lines operate in push-pull mode.

SDIO_CMD has two modes of operation:

- Open-circuit mode for initialization (only for MMC version V3.31 or earlier)
- Push-pull mode for command transfer (SD/SD I/O cards and MMC V4.2 also use push-pull drivers during initialization).

SDIO uses two clock signals:

- SDIO adapter clock (SDIOCLK=HCLK)
- AHB bus clock (HCLK/2)

The following table applies to the MultiMediaCard/SD/SD I/O card bus:

Table 22-1 SDIO Pin Definition

Pin	Direction	Description
SDIO_CK	Output	Multimedia card/SD/SDIO card clock. This is the clock line from the host to the card
SDIO_CMD	Bidirectional	Multimedia card/SD/SDIO card commands. This is a bidirectional command/response signal line
SDIO_D[7:0]	Bidirectional	Multimedia card/SD/SDIO card data. These are bidirectional data buses.

22.2.1. SDIO Adapter

The SDIO adapter includes a control unit, a command unit, and a data unit, and can generate signals to the card. These signals are described in detail as follows:

SDIO_CLK: The clock provided to the card by the SDIO controller. Each clock cycle sends one bit of command or data directly on the command line (SDIO_CMD) and all data lines (SDIO_DAT). The SDIO_CLK frequency can be between 0 MHz and 20 MHz for MMC card version V3.31, between 0

MHz and 48 MHz for MMC card version V4.2, and between 0 MHz and 25 MHz for SD or SD I/O cards.

SDIO_CMD: This signal is a bidirectional command channel for card initialization and command transmission. 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 modes of operation: open-drain mode for initialization (only for MMC cards V3.31 and earlier) and push-pull mode for command transmission (SD cards/SD I/O cards and MMC cards version 4.2 are also initialized with push-pull mode).

SDIO_DAT[7:0]: These signal lines are bidirectional data channels. The data signal lines operate in push-pull mode. Only the card or host will drive these signals each time. By default, only DAT0 is used for data transfer after power up or reset. The SDIO adapter can be configured with a wider data bus for data transfer, using DAT0-DAT3 or DAT0-DAT7 (for MMCV4.2 only). SDIO has internal pull-ups for data signal lines DAT1-DAT7. Upon entering 4-bit mode, the card disconnects the internal pull-ups for DAT1 and DAT2 (the DAT3 internal pull-up remains unchanged due to the use of the CS chip select in SPI mode). Correspondingly, upon entering 8-bit mode, the internal pull-ups for DAT1, DAT2, and DAT4-DAT7 are disconnected.

The SDIO adapter is the interface to SD/SD I/O/MMC/CE-ATA and consists of 3 subunits:

22.2.1.1. Control unit

The control unit contains power management function and clock management function for memory card clock. Power management is controlled by the SDIO_PWRCTL register, which implements power down and power up. The power saving mode is configured by setting the PWRSAV bit of SDIO_CLKCR to enable SDIO_CLK to be turned off when the bus is idle. clock management generates the SDIO_CLK clock signal to the card.

22.2.1.2. Command unit

The Command Unit implements sending and receiving commands to and from the card. The data transfer flow is controlled by the Command State Machine (CSM). Command transmission starts after a write operation to the SDIO_CMD register and setting the start_cmd bit of this register to 1. First, a command is sent to the card, which contains 48 bits and is sent through the SDIO_CMD line, one bit of data per SDIO_CLK. This 48-bit command contains 1-bit start bit, 1-bit transfer bit, 6-bit command index (defined by the CMDINDEX bit of the SDIO_CMD register), 32-bit parameters (defined by SDIO_CMDARG), 7-bit CRC and 1-bit stop bit. The response from the card is then received (in case the CMDINDEX bit of the SDIO_CMD register is not 0b00 or 0b10) and is divided into a short response of 48 bits and a long response of 136 bits, both of which are present in the SDIO_RESP0 - SDIO_RESP3 registers.

Command Channel State Machine (CPSM)

When the command register is written and the enable bit is set, sending of commands begins. When command sending is completed, the command channel state machine (CPSM) sets the status flag and enters the idle state when no response is required (see the figure below). When a response is received, the received CRC code will be compared with the internally generated CRC code and then the corresponding status flag will be set.

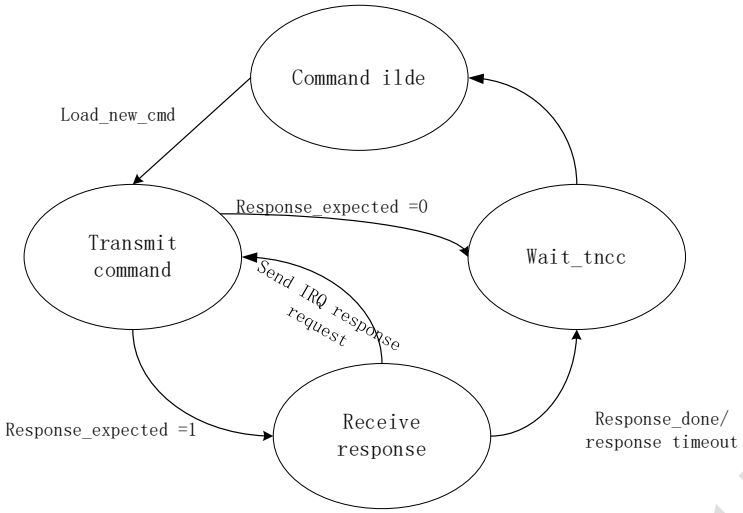


Figure 22-2 SD_MMC Command Channel State Machine (CPSM)

The command path state machine performs the following functions based on the command register bit values:

1. Send_initialization - An initialization sequence of 80 clocks is sent before the command is sent.
2. Response_expected - The expected response to the command. After the command is sent, the command path state machine receives a 48-bit or 136-bit response to send to the SDIO adapter. If the start bit of the card response is not received within the number of clocks programmed in the timeout register, the response timeout and command completion bits are set in the interrupt status register. If the response anticipation bit is not set, the command completion bit is set in the interrupt status register after the command is issued.
3. Response_length - If this bit is set, a 136-bit response is received; if it is not set, a 48-bit response is received.
4. Check_response_crc - If this bit is set, the command path compares the CRC7 received in the response with the internally generated CRC7. If the two do not match, the response CRC error signal is sent to the BIU; that is, the response CRC error bit is set in the raw interrupt status register.

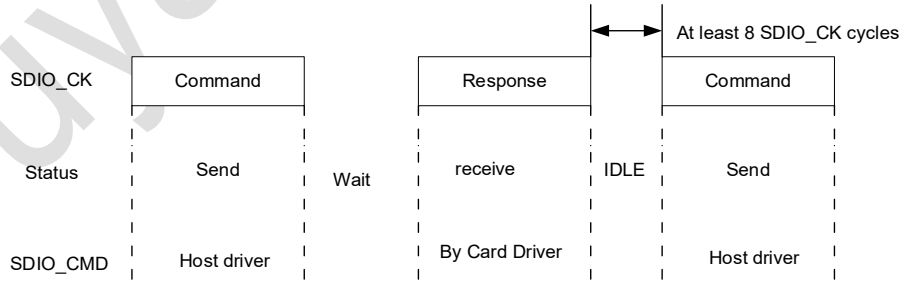


Figure 22-3 SDIO command transmission

Command Format

Command: A command is used to start an operation. The host issues a command with an address or a broadcast command to a specified card or to all cards (broadcast commands are only available for MMC V3.31 or earlier). Commands are transmitted serially on the CMD line. The length of all commands is fixed at 48 bits. The following table gives the general command format on the MMC, SD memory cards and SDIO cards.

The CE-ATA command is an extension of the MMC V4.2 command, so it has the same format.

The command channel is in half-duplex mode, so that commands and responses can be sent and received separately. If the CPSM is not in the transmit state, the SDIO_CMD output is in the high resistance state. The data on SDIO_CMD is synchronized with the rising edge of SDIO_CK.

Table 22-2 command format

Bit	Width	Value	Description
47	1	0	Start bit
46	1	1	Transfer bit
[45:0]	6	-	Command Index
[39:8]	32	-	Parameter
[7:1]	7	-	CRC7
0	1	1	End bit

Response: The response is sent from a card with the specified address to the host, for all cards in MMC V3.31 or previous versions

A response is sent simultaneously; a response is an answer to a previously received command. The response is transmitted serially on the CMD line.

SDIO supports 2 response types, both with CRC error detection:

- 48-bit short response
- 136-bit long response

If the response does not contain a CRC (as in the case of CMD1 response), the device driver should ignore the CRC failure status.

Table 22-3 Short response format

Bit	Width	Value	Description
47	1	0	Start bit
46	1	1	Transfer bit
[45:0]	6	-	Command Index
[39:8]	32	-	Parameter
[7:1]	7	-	CRC7
0	1	1	End bit

Table 22-4 Long Response Format

Bit	Width	Value	Description
135	1	0	Start bit
134	1	0	Transfer bit
[133:128]	6	111111	Reserved
[127: 1]	127	-	CID or CSD (includes internal CRC7)
0	1	1	End bit

The command register contains the command index (the 6 bits sent to the card) and the command type; the command itself determines whether a response is required and the type of response

The command itself determines whether a response is required and the type of response, 48 bits or 136 bits. The status flags in the command channel are shown in the following table.

Table 22-5 Command channel status flag

Symbol	Description
CMDREND	The CRC of the response is correct
CCRCFAIL	The CRC of the response is incorrect
CMDSENT	Command (command that do not require a response) have been sent
CTIMEOUT	Response timeout
CMDACT	Command being sent

The CRC generator calculates the CRC checksum of all bits preceding the CRC code, including the start bit, transmit bit, command index, and command parameters (or card status). For the long response format, the CRC checksum is calculated for the first 120 bits of the CID or CSD; note that the start bit, transmit bit, and six reserved bits in the long response format are not involved in the CRC calculation.

The CRC checksum is a 7-bit value of:

22.2.1.3. Data unit

The data unit implements the data transfer between the host and the card. When the data width is 8 bits, the SDIO_DAT[7:0] signal line is used for data transfer; when the data width is 4 bits, the SDIO_DAT[3:0] signal line is used for data transfer; when the data width is 1 bit, the SDIO_DAT[0] signal line is used for data transfer. The data transfer flow is controlled by the data state machine (DSM).

- **Data Transmission State Machine (DTSM)**

The DTSM operates at the SDIO_CK frequency, and the card bus signal is synchronized with the rising edge of SDIO_CK.

As shown in the figure,the data transmitter state machine starts data transmission two clocks after receiving a response to a data write command; this happens even if the command path state machine detects an error or a response CRC error. If a response is not received from the card due to a response timeout, no data is transmitted. Depending on the value of the trans-fer_mode bit in the command register, the data sending state machine places the data on the card data bus as a stream or a block.

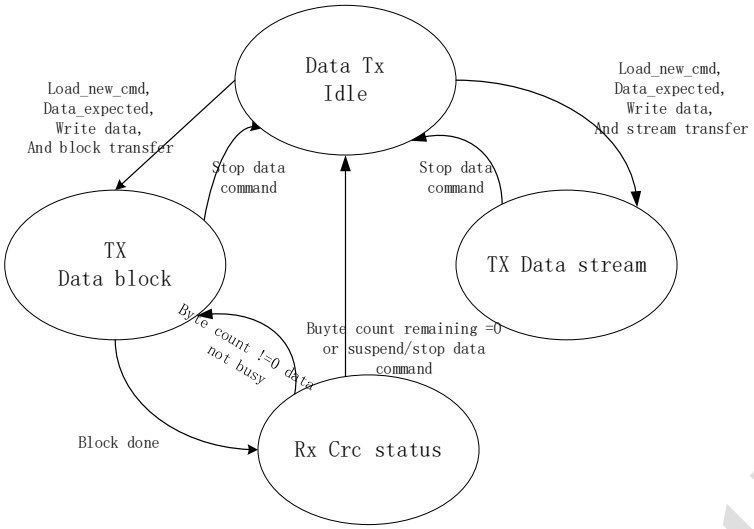


Figure 22-4 Data Transmission State Machine (DTSM)

- **Data Receiving State Machine (DRSM)**

As shown in the figure, the data receive state machine receives data two clock cycles after the end bit of the data read command, even if the command path detects a response error or a response CRC error. If no response is received from the card because of a response timeout, the SDIO adapter does not receive a signal that the data transfer is complete; this happens if the command sent is an illegal operation on the card, which prevents the card from starting to read the data transfer. If data is not received before the data timeout, the data receive state machine signals the SDIO adapter to time out the data and ends the data transfer. Depending on the value of the transfer_mode bit in the command register, the data receive state machine fetches data from the card data bus as a stream or a block.

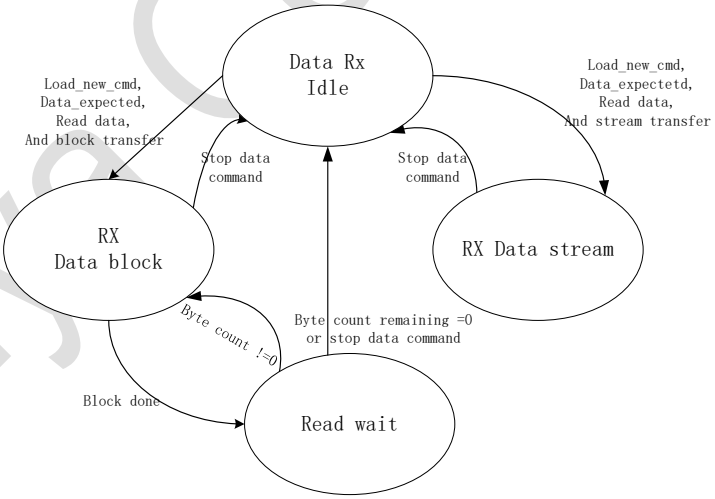


Figure 22-5 data-receiving state machine（DRSM）

22.2.1.4. SDIO clock control

There are three register bits that can be used for clock processing, namely the CKSEL, SMPEN, and SMPCLKSEL bits of the clk:

1. CKSEL: used to control the output command, data selection
2. SMPEN: pre-sampling enable, SMPCLKSEL: clock phase selection, you can select the specific phase of the clock

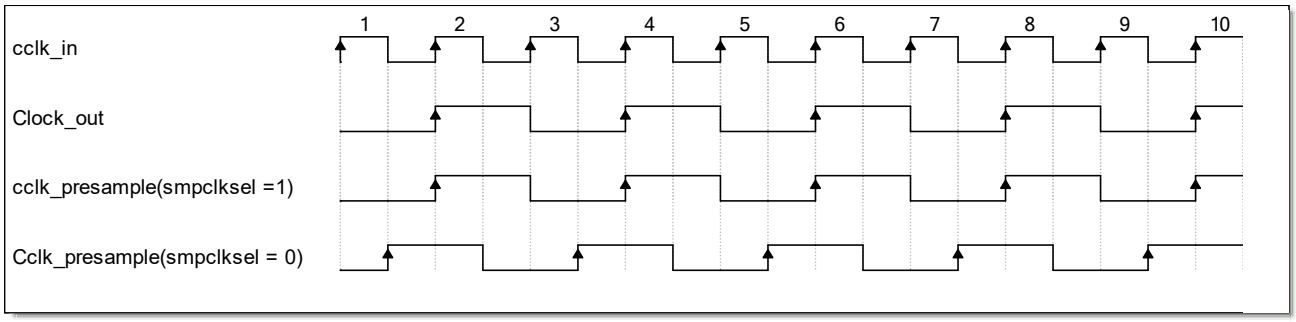


Figure 22-6 Schematic diagram of clk_presample

22.2.2. SDIO AHB interface

The AHB interface generates interrupt and DMA requests and accesses the SDIO interface registers and data FIFO. it contains a data channel, register decoder, and interrupt/DMA control logic.

22.2.2.1. SDIO interrupt

The interrupt control logic generates an interrupt request when at least one of the selected status flags is high. There is a mask register for selecting the conditions under which an interrupt can be generated, and if the corresponding mask flag is set, the corresponding status flag can generate an interrupt.

22.2.2.2. SDIO/DMA interface: Data transfer between SDIO and memory

SDIO/DMA: The process of transferring data between SDIO and memory.

In the following example, the host controller uses CMD24 (WRITE_BLOCK) to transfer 512 bytes from the host to the MMC card, and the DMA controller is used to fill the data from the memory to the FIFO of the SDIO.

1. Perform the card identification process
2. Increase SDIO_CK frequency
3. send CMD7 command to select the card
4. Configure the DMA as follows 2.
   a) Enable DMA2 controller and clear all interrupt flags
   b) Set the source address register of DMA2 channel 4 to the base address of the memory buffer, and the destination address register of DMA2 channel 4 to the address of the SDIO_FIFO register
   c) Set the DMA2 channel 4 control register (memory increment, non-peripheral increment, data width of peripheral and source is word width)
   d) Enable DMA2 channel 4
5. CMD24 (WRITE_BLOCK) with the following operations:
   a) Set SDIO data length register (SDIO data clock register should be set before executing the card identification process)
   b) Set the SDIO parameter register to the address of the data to be transmitted in the card

c) Set SDIO command register: CmdIndex set to 24 (WRITE_BLOCK); WaitRest set to 1 (SDIO card host waiting for response); start_cmd set to 1 (enable SDIO card host to send command), keep other fields as their reset value.

d) Wait for CD interrupt, then set SDIO data register: DTEN set to 1 (enable SDIO card host to send data); DTDIR set to 0 (controller to card direction); DTMODE set to 0 (block data transfer); DMAEN set to 1 (enable DMA); DBLOCKSIZE set to 9 (512 bytes); other fields do not need to be set.

e) Wait for SDIO_STA[10]=DBCKEND.

6. Query enable status register of the DMA channel to confirm that no channel is still in the enable state.

22.2.3. Card Function Description

22.2.3.1. Card Register

The card internally defines the interface registers: OCR, CID, CSD, EXT_CSD, RCA, DSR and SCR. these registers can only be accessed by the corresponding commands. the OCR, CID, CSD and SCR registers contain some card-specific information, while the RCA and DSR registers are configuration registers that store the actual configuration parameters. The ECSD registers contain both card-specific information and the actual configuration parameters. For specific information, please refer to the relevant specification.

OCR register: The 32-bit Operating Conditions Register (OCR) stores the card's VDD voltage description and access mode indication (MMC). In addition, this register includes a status information bit. This status bit is set if the card power-up process has been completed. This register is a little different between MMC and SD cards. The host can use CMD1 (MMC), ACMD41 (SD memory card), and CMD5 (SD I/O) to access the contents of this register.

CID Register: The Card Identification Register (CID) is 128 bits wide. It contains the card identification information used during the card identification phase. Each read/write (RW) card should have a unique identification number. The host can get the contents of this register using CMD2 and CMD10.

CSD Register: The Card Specific Data Register provides access to information about the contents of the card. The CSD defines the data format, error correction type, maximum data access time, data transfer speed, availability of the DSR registers, etc. The programmable portion of the registers can be modified using CMD27. The host can use CMD9 to get the contents of this register.

RCA register: The writable 16-bit relative card address register holds the card address, which is published outward by the card during card initialization. This address is used for communication between the addressed host and the card after the card identification process. The host can use CMD3 to request that the card issue a new relative address (RCA).

Note: The default value of the register for the RCA is 0x0001 (MMC) or 0x0000 (SD/SD I/O). This value is reserved and is used to set all cards to the standby (Stand-by) state via CMD7.

DSR register (optional): The 16-bit drive phase register is optional and can be used to improve bus performance in extended operating conditions (depending on parameters like bus length, transfer rate and number of cards). The CSD register contains information about the DSR register usage.

The default value of the DSR register is 0x404. The host can use CMD4 to get the contents of this register.

SCR register: Only SD/ SD I/O (if there is a memory module) has this register. In addition to the CSD register, there is another configuration register named SD Card Configuration Register (SCR) in addition to the CSD register: Which is only used for SD cards. The SCR provides information about the special features that are configured to a specific SD memory card. The size of the SCR register is 64 bits. This register should be set by the SD memory card manufacturer before shipping. The host can use ACMD51 to get the contents of this register.

The response format R1 contains a 32-bit card status field that is used to send card status information to the card host (it is possible that this information is present in the local status register). Unless otherwise specified, the status returned by the card is always associated with the previous command.

Table 4-2 defines the different status messages. The abbreviated definitions of the relevant type and clear condition fields in the table are as follows:

Type:

- E: Error bit
- S: Status bit
- R: Detection bit, and set according to the actual command response
- X: Detect bit, and is set during the command execution. The SDIO card host queries the status of the card by sending a status command to read out these bits.

Clear conditions:

- A: Based on the current state of the card
- B: Always related to the previous command. Cleared by receiving the correct command (with a command delay).
- C: Read to clear

Table 22-6 Card Status

Bit	Name	Type	Value	Description	Clear condition
31	ADDRESS_OUT_OF_RANGE	E R X	'0' = No error '1' = Error	A multi-block or stream read/write operation (even if starting from a legal address) attempts to read or write more than the capacity of the card.	C
30	ADDRESS_MISALIGN		'0' = No error '1' = Error	The first data block defined by the address parameter in the command (against the current data block length) is not aligned with a physical block of the card. A multi-block or stream read/write operation (even if starting from a legal address)	C

Bit	Name	Type	Value	Description	Clear condition
				attempts to read or write a block of data that is not aligned with a physical block.	
29	BLOCK_LEN_ERROR		'0' = No error '1' = Error	The parameter of the SET_BLOCKLEN command exceeds the maximum allowed range of the card, or the previously defined block length is illegal for the current command (e.g., the host issues a write command and the current block length is less than the minimum allowed by the card, while not allowing to write part of the block).	C
28	ERASE_SEQ_ERROR		'0' = No error '1' = Error	The erase command was sent in the wrong order.	C
27	ERASE_PARAM	EX	'0' = No error '1' = Error	An illegal erase group was selected for the erase.	C
26	WP_VIOLATION	E X	'0' = No error '1' = Error	Attempt to program a write-protected data block.	C
25	CARD_IS_LOCKED	S R	'0' = Card unlocked '1' = Card is locked	When this bit is set, it indicates that the card is locked.	A
24	LOCK_UNLOCK_FAILED	E X	'0' = No error '1' = Error	Wrong order of commands in locking/unlocking or wrong password detected	C
23	COM_CRC_ERROR	E R	'0' = No error '1' = Error	CRC checksum error in the previous command.	B
22	ILLEGAL_COMMAND	E R	'0' = No error '1' = Error	For the current card status, the command is illegal.	B
21	CARD_ECC_FAILED	E X	'0' = Success '1' = Failure	The card's internal ECC checksum was implemented, but failed in correcting the data.	C
20	CC_ERROR	E R	'0' = No error '1' = Error	(Not defined in the standard) An error occurred inside the card, independent of the host's command.	C
19	ERROR	E X	'0' = No error '1' = Error	Generated a (not defined in the standard) related to the execution of the previous host command Card internal error	C
18:17	Reserved				

Bit	Name	Type	Value	Description	Clear condition
16	CID/CSD_OVERWRITE	E X	'0' = No error '1' = Error	It can be any of the following errors: The CID register has already been written and cannot be overwritten The read-only portion of the CSD does not match the contents of the card Attempted copy or permanent write-protected reverse operation, i.e. restoring the original or unprotecting the write.	C
15	WP_ERASE_SKIP	E X	'0' = Unprotected '1' = Protected	Only part of the address space is erased when a write-protected data block is encountered that already exists	C
14	CARD_ECC_DISABLED	S X	'0' = Not allowed '1' = Allowed	The command is executed without using the internal ECC.	A
13	ERASE_RESET		'0' = Unprotected '1' = Protected	The sequence into the erase process is aborted because a command other than the erase sequence (not a CMD35, CMD36, CMD38, or CMD13 command) is received.	C
12:9	CURRENT_STATE	S R	0 = Idle 1 = Ready 2 = Recognized 3 = Standby 4 = Transmit 5 = Data 6 = Receive 7 = Program 8 = Disconnect 9 = Busy test 10 to 15 = Hold	The state of the card's state machine when the command is received. If the execution of a command causes a change in the state of B to a change in state, this change will be reflected in the response to the next command This change will be reflected in the response to the next command. These four bits are interpreted as decimal numbers 0 through 15.	B

Bit	Name	Type	Value	Description	Clear condition
8	READY_FOR_DATA	S R	'0' = Not ready '1' = Ready	Corresponds to the signal of an empty buffer on the bus.	
7	SWITCH_ERROR	E X	'0' = No error '1' = Error	The card is not converted to the desired mode as required by the SWITCH command.	B
6	Reserved				
5	APP_CMD	S R	'0' = Not allowed '1' = Allowed	Card expects ACMD, or indicates that the command has been interpreted as an ACMD command	C
4	Reserved for SD I/O cards				
3	AKE_SEQ_ERROR	E R	'0' = No error '1' = Error	There is an error in the order of validation	C
2	Reserved for application-related commands.				
1:0	Test patterns reserved for manufacturers				

22.2.3.2. SD Status Register

The SD status contains status bits related to specific functions of the SD memory card and some status bits related to future applications. The length of the SD status is a 512-bit block of data. Upon receipt of an ACMD13 command (CMD55, then CMD13), the contents of this register are transferred to the SDIO card host. The ACMD13 command can only be sent when the card is in the transmit state (the card has been selected).

Table 4-3 defines the different SD status register information. The abbreviations in the table regarding the type and clear condition fields define the following types:

- E: Error bit
- S: Status bit
- R: Detection bit, and is set according to the actual command response
- X: Detect bit, set during command execution. The SDIO card host queries the status of the card by sending status commands to read these bits.

Clear conditions:

- A: Based on the current status of the card
- B: Always related to the previous command. Cleared by receiving the correct command (with a delay of one command).
- C: Read to clear

Table 22-7 SD Status

Bit	Name	Type	Value	Description	Clear condition
511:510	DAT_BUS_WIDTH	S R	'00' = 1 (Default) '01' = Reserved '10' = 4 bits wide '11'= Reserved	The current data bus width as defined by the SET_BUS_WIDTH command.	A
509	SECURED_MODE	S R	'0' = Not in confidential mode '1' = In confidential mode	Card is in secure mode of operation '0' = Not in secure mode '1' = In secure mode	A
508:496	Reserved				
495:480	SD_CARD_TYPE	S R	00xxh' = In physical specification version 1.01 A ~2.00 of the SD memory card ('x' indicates any value). Defined cards are: '0000' = Universal SD read/write card '0001' = SD ROM card	The lower 8 bits of this field can be used to define different variants of SD memory cards in the future (each bit can be used to define a different SD type). The high 8 bits can be used to define SD cards that do not adhere to the current SD physical layer specification	A
479:448	SIZE_OF_PROTECTED_AREA	S R	Size of the protected area (See description below)	(See description below)	A
447:440	SPEED_CLASS	S R	Type of speed of the card (See description below)	(See description below)	A
439:432	PERFORMANCE_MOVE	S R	Transfer performance in 1MB/sec (See description below)	(See description below)	A
431:428	AU_SIZE	S R	AU size (See description below)	(See description below)	A
427:424	Reserved				
423:408	ERASE_SIZE	S R	Number of AU's that can be erased at one time	(See description below)	A
407:402	ERASE_TIMEOUT	S R	Erase the timeout value for the range specified by UNIT_OF_ERASE_A	(See description below)	A
401:400	ERASE_OFFSET	S R	Fixed offset value added during erase	(See description below)	A
399:312	Reserved				
311:0	Reserved for manufacturers				

SIZE_OF_PROTECTED_AREA

The way to set this bit is different for standard capacity cards and high capacity cards. For standard capacity cards, the capacity of the protected area is calculated by the following formula:

Protected area = SIZE_OF_PROTECTED_AREA * MULT * BLOCK_LEN

The unit of SIZE_OF_PROTECTED_AREA is MULT * BLOCK_LEN.

For high capacity cards, the capacity of the protected area is calculated by the following equation:

Protected area = SIZE_OF_PROTECTED_AREA

The unit of SIZE_OF_PROTECTED_AREA is bytes.

SPEED_CLASS

These 8 bits indicate the type of speed and can be calculated by calculating the value of PW/2 (PW is the write performance).

Table 22-8 Speed Type Code

SPEED_CLASS	Numerical Definition
00h	Type 0
01h	Type 2
02h	Type 4
03h	Type 6
04h-FFH	Reserved

PERFORMANCE_MOVE

These 8 bits indicate the mobile performance (Pm) in 1MB/sec. If the card does not use RU (units of record) to move data, Pm should be considered to be infinite. Setting this field to FFh indicates infinity.

Table 22-9 Speed Type Code

SPEED_CLASS	Numerical Definition
00h	Undefined
01h	1MB/S
02h	2MB/S
……	……
Feh	254MB/S
FFh	Infinity

AU_SIZE

These 4 bits indicate the length of the AU, and the value is a multiple of 16K bytes to the power of 2.

Table 22-10 AU_SIZE code

SPEED_CLASS	Numerical Definition
00h	Undefined
01h	16KB
02h	32KB
03h	64KB
04h	128KB
05h	256KB
06h	512KB

07h	1MB
08h	2MB
09h	4MB
AH-Fh	Infinity

Depending on the capacity of the card, the maximum AU length is defined in the table below. The card can have any AU length between the RU length and the maximum AU length.

Table 22-11 Maximum AU length

Capacity	**16MB-64MB**	**128MB-256MB**	**512MB**	**1GB-32GB**
Maximum AU length	512KB	1MB	2MB	4MB

ERASE_SIZE

This 16-bit field gives the NERASE, and ERASE_TIMEOUT defines the timeout when NERASE AU's are erased. The host should determine the appropriate number of AU's to be erased in one operation so that the host can display the progress of the erase operation. If this field is 0, the timeout calculation for erase is not supported.

Table 22-12 ERASE_SIZE code

ERASE_CLASS	**Numerical Definition**
0000h	Timeout operation for erasure is not supported
0001h	1 AU
0002h	2 AU
0003h	3 AU
……	……
FFFFh	65535 AU

ERASE_TIMEOUT

These 6 bits give the TERASE, ERASE_SIZE indicates the number of AU's to be erased, this value gives the erase timeout from the offset. The range of ERASE_TIMEOUT can be defined up to a maximum of 63 seconds, and the card manufacturer can choose the appropriate combination of ERASE_SIZE and ERASE_TIMEOUT according to the specific implementation by first determining ERASE_TIMEOUT and then ERASE_SIZE.

Table 22-13 Erase timeout code

ERASE_TIMEOUT	**Numerical Definition**
00	Timeout operation for erasure is not supported
01	1 second
02	2 second
03	3 second
……	……
63	63 second

ERASE_OFFSET

These 2 bits give the TOFFSET, and this value is meaningless when ERASE_SIZE and ERASE_TIMEOUT are the same as 0.

Table 22-14 Erase Offset Code

ERASE_OFFSET	Numerical Definition
0	0 second
1	1 second
2	2 second
3	3 second

22.2.3.3. I/O mode of SD

I/O interrupt for SD

In order for the SD I/O card to interrupt the multimedia card/SD module, there is a pin with interrupt function on the SD interface - pin 8. In 4-bit SD mode this pin is SDIO_D1, which the card uses to make an interrupt request to the multimedia card/SD module. For each card or function within the card, the interrupt function is optional. The interrupts for SD I/O are level valid, i.e. the interrupt signal line must remain valid (low) until it is recognized and gets a response from the multimedia card/SD module, and remain invalid (high) after the interrupt process is completed. After the interrupt request is serviced by the multimedia card/SD module, the interrupt status bit is cleared by an I/O write operation that writes the appropriate bit to the SD I/O card's internal register. The interrupt outputs of all SD I/O cards are active low and pull-up resistors are provided on all data lines (SDIO/D[3:0]) of the multimedia card/SD module. The multimedia card/SD module samples pin 8 (SDIO_D/IRQ) and performs interrupt detection during the interrupt phase, and the value on this signal line is ignored at other times.

Both memory operations and I/O operations have interrupt phases, and the interrupt phase definition for a single data block operation is different from the interrupt phase definition for multiple data block transfer operations.

I/O pause and resume for SD

In a multifunction SD I/O card or a card with both I/O and memory functions, multiple devices (I/O and memory) share the MMC/SD bus. In order to enable multiple devices in the MMC/SD module to share the bus, SD I/O cards and composite cards can optionally implement the concept of suspend/resume; if a card supports suspend/resume, the MMC/SD module can temporarily stop the data transfer operation (suspend) of a function or memory, thereby giving the bus to other functions or memories with higher priority, and then resume the original transfer after this higher priority is completed. After this higher priority transfer is completed, the original suspended transfer is resumed. Support for suspend/resume operations is optional. The following steps are used to perform a suspend/resume operation on the MMC/SD bus:

1. Determine the current function of the SDIO_D[3:0] signal line
2. Request a low priority or slow operation pause
3. Wait for the pause operation to complete and confirm that the device is paused

4. Start high priority transmission
5. Wait for the high priority transmission to finish
6. Resume the suspended operation

SD I/O ReadWait

The optional Read Wait (RW) operation is only available for SD cards in 1-bit or 4-bit mode. The Read Wait operation allows the MMC/SD module to request a card to temporarily stop data transfer while it is reading multiple registers (IO_RW_EXTENDED, CMD53), while allowing the MMC/SD module to send commands to other functions in the SD I/O device. To determine if a card supports the read-wait protocol, the MMC/SD module should detect the card's internal registers. The duration of the read wait is related to the interrupt phase.

22.2.4. Command

22.2.4.1. Application-related commands and general commands

The SD card host module system is used to provide a standard interface for a wide range of application types, but with user- and application-specific functionality. Therefore, two types of generic commands are defined in the standard: application-related commands (ACMD) and generic commands (GEN_CMD).

When the card receives an APP_CMD (CMD55) command, the card expects the next command to be an application-related command. The application-related command (ACMD) has the same format structure as a normal multimedia card and can use the same CMD number because it appears after APP_CMD(CMD55), so the card recognizes it as an ACMD command. If following APP_CMD(CMD55) is not an already defined application-related command, it is considered a standard command; example: there is an SD_STATUS(ACMD13) application-related command, if CMD13 is received immediately after APP_CMD(CMD55), it is interpreted as SD_STATUS(ACMD13); however if the card receives CMD7 immediately after APP_CMD(CMD55) and this card does not have ACMD7 defined, it is interpreted as a standard CMD7(SELECT/DESELECT_CARD) command.

To use the manufacturer's custom ACMD, the SD card host needs to do the following:

1. Send the APP_CMD (CMD55) command

   The card sends back a response to the multimedia/SD card module indicating that the APP_CMD bit is set and waiting for the ACMD command.

2. Send the specified ACMD

   The card sends back a response to the multimedia/SD card module indicating that the APP_CMD bit is set and the received command has been correctly parsed according to the ACMD command; if a non-ACMD command is sent, the card will be processed as a normal multimedia card command while clearing the APP_CMD bit in the status register of the card.

If an illegal command (either ACMD or CMD) is sent, it will be handled incorrectly as a standard illegal multimedia card command.

The bus operation procedure of GEN_CMD command is the same as single data block read/write command (WRITE_BLOCK, CMD24 or READ_SINGLE_BLOCK, CMD17); then the parameter of the

command indicates the direction of data transfer instead of address, and the data block has user-defined format and meaning.

Before sending the GEN_CMD (CMD56) command, the card must be selected (the state machine is in the transmission state) and the length of the data block is defined by SET_BLOCKLEN (CMD16). The response to the GEN_CMD(CMD56) command is in R1b format.

22.2.4.2. Command Type

There are four different types of application-related and generic commands:

1. Broadcast command (BC): sent to all cards, no response returned.
2. Broadcast command with response (BCR): sent to all cards, while receiving a response returned from all cards.
3. Command with Addressing (Point-to-Point) (AC): sent to the selected card and does not include data transfer on the SDIO_D signal line.
4. Data transfer command (AC) with addressing (point-to-point): sent to the selected card and includes data transfer on the SDIO_D signal line.

22.2.4.3. Command Format

Commands for the multimedia card/SD card module

Table 22-15 Block transfer based write command

CMD Index	Type	Parameter	Response Format	Preparation	Description
CMD23	ac	[31:16] = 0 [15:0] = Dumber of data blocks	R1	SET_BLOCK_COUNT	Defines the number of blocks to be transferred in a subsequent multi-block read or write command
CMD24	adtc	[31:0] = Data address	R1	WRITE_BLOCK	Write a block of the length selected by the SET_BLOCKLEN command
CMD25	adtc	[31:0] = Data address	R1	WRITE_MULTIPLE_BLOCK	Write data blocks continuously until a STOP_TRANSMISSION command is received or the specified number of blocks has been reached
CMD26	adtc	[31:0] = Filling bit	R1	PROGRAM_CID	Program the card's identification register. This command can be sent only once for each card. There is a hardware mechanism in the card to prevent multiple programming operations. Usually this command is reserved for the manufacturer
CMD27	adtc	[31:0] = Filling bit	R1	PROGRAM_CSD	Program the programmable bits in the card's CSD.
CMD28	ac	[31:0] = Data address	R1b	SET_WRITE_PROT	If the card has write protection feature, this command sets the write protection bit of the

CMD Index	Type	Parameter	Response Format	Preparation	Description
					specified group. The write protect feature is set in the special data area of the card (WP_GRP_SIZE).
CMD29	ac	[31:0] = Data address	R1b	CLR_WRITE_PROT	If the card is write-protected, this command clears the write-protect bits of the specified group
CMD30	adtc	[31:0] = Write protect data address	R1	SEND_WRITE_PROT	If the card is write-protected, this command asks the card to send the status of the write-protect bit
CMD31	Reserved				

Table 22-16 Block transfer-based write protect command

CMD Index	Type	Parameter	Response Format	Preparation	Description
CMD28	ac	[31:0] = Data block address	R1b	SET_WRITE_PORT	Defines the number of blocks to be transferred in a subsequent multi-block read or write command
CMD29	ac	[31:0] = Data address	R1b	CLR_WRITE_PORT	Write a block of the length selected by the SET_BLOCKLEN command
CMD30	adtc	[31:0] = Write protect data address	R1	SEND_WRITE_PROT	If the card is write-protected, this command asks the card to send the status of the write-protect bit
CMD31	Reserved				

Table 22-17 Erase command

CMD Index	Type	Parameter	Response Format	Preparation	Description
CMD32 … CMD34	Reserved. For backward compatibility with older versions of the pair media card protocol, these command codes cannot be used.				
CMD35	ac	[31:0] = Data address	R1b	CLR_WRITE_PORT	Write a block of the length selected by the SET_BLOCKLEN command

CMD Index	Type	Parameter	Response Format	Preparation	Description
CMD36	adtc	[31:0] = Write protect data address	R1	SEND_WRITE_PROT	If the card is write-protected, this command asks the card to send the status of the write-protect bit
CMD37	Reserved. For backward compatibility with older versions of the pair media card protocol, this command code cannot be used				
CMD38	ac	[31:0]=Filler	R1	ERASE	Erase the previously selected data block

Table 22-18 I/O mode command

CMD Index	Type	Parameter	Response Format	Preparation	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. This command specifies a card and register, and if the write flag is set also provides the data to be written. The R4 response contains the data read from the specified register. This command accesses application-related registers that are not defined in the multimedia card standard.
CMD40	bcr	[31:0] = Write protect data address	R5	GO_IRO_STATE	Places the system in interrupt mode.
CMD41	Reserved				

Table 22-19 Lock command

CMD Index	Type	Parameter	Response Format	Preparation	Description
CMD42	adtc	[31:0] = Filling bit	R1b	LOCK_UNLOCK	Set/Clear the password or locks/unlocks the card. The length of the data block is set by the SET_BLOCKLEN command.
CMD43 …… CMD54	Reserved				

Table 22-20 Use the relevant command

CMD Index	Type	Parameter	Response Format	Preparation	Description
CMD55	ac	[31:16] = RCA [15：0] = Filling bit	R1	APP_CMD	Indicates that the next command on the card is an application-related command rather than a standard command
CMD56	adtc	[31:1] = Filling bit [0] = RD/WR			Either for transferring a data block to the card or for reading a data block from the card in a general-purpose or application-related command. The length of the data block is set by the SET_BLOCKLEN command.
CMD57 … CMD59	Reserved				
CMD60 …… CMD63	Reserved for manufacturers				

22.2.5. Response

All responses are transmitted on the SDIO_CMD signal line via the MCCMD command. The transmission of a response always starts from the leftmost side of the bit string corresponding to the response word, and the length of the response word is related to the type of response.

A response always has a start bit (always 0) that follows the direction bit of the transmission (card = 0). The value labeled x in the table below indicates a variable portion. All responses are CRC protected, except for the R3 response type. Each command code word has an end bit (always 1).

There are seven response types, and their formats are defined as follows:

22.2.5.1. R1 (general response command)

Code length = 48 bits. Bits 45:40 indicate the index of the command to respond to, which has a value between 0 and 63. The status of the card is encoded by 32 bits.

Table 22-21 Response

Bit	Domain width	Numerical value	Description
47	1	0	Start bit
46	1	0	Transfer bit
[45:40]	6	X	Command Index
[39:8]	32	X	Card Status
[7:1]	7	X	CRC7
0	1	1	End bit

22.2.5.2. R1b

Same format as R1, but with the option to send a busy signal on the data line. Upon receipt of these commands, the card may become busy depending on the status prior to receipt of the command.

22.2.5.3. R2 (CSD, CSD register)

Code length = 136 bits. The contents of the CID register will be issued as the response of CMD2 and CMD10. The contents of the CSD register will be sent out as a response to CMD9. The card sends only bits [127...1] of CID and CSD, and bit 0 of these registers is replaced by the end bit of the response at the receiving end. The card indicates that it is performing an erase operation by pulling MCDAT low; the actual erase operation may take so long that the host can send a CMD7 command to not select this card.

Table 22-22 Response

Bit	Domain width	Numerical value	Description
135	1	0	Start bit
134	1	0	Transfer bit
[133:128]	6	‘111111’	Command Index
[127:1]	127	X	Card Status
0	1	1	End bit

22.2.5.4. R3 (OCR register)

Code length = 48 bits. the content of the OCR register will be issued as a response to CMD1. The level code is defined as: limited voltage window = low, card busy = low.

Table 22-23 R3 Response

Bit	Domain width	Numerical value	Description
47	1	0	Start bit
46	1	0	Transfer bit
[45:40]	6	‘111111’	Reserved
[39:8]	32	X	OCR register
[7:1]	7	‘111111’	Reserved
0	1	1	End bit

22.2.5.5. R4 (Fast I/O)

Code length = 48 bits. The parameter field contains the RCA of the specified card, the address of the register to be read or written, and its content.

Table 22-24 R4 Response

Bit		Domain width	Numerical value	Description
47		1	0	Start bit
46		1	0	Transfer bit
[45:40]		6	'111111'	Reserved
[39:8] Parameter field	[31:16]	16	X	RCA
	[15:8]	8	X	Register Address
	[7:0]	8	X	Read the contents of the register
[7:1]		7	'111111'	CRC7
0		1	1	End bit

22.2.5.6. R4b

Suitable for SD I/O cards only: an SDIO card will return a unique SDIO response R4 upon receipt of CMD5.

Table 22-25 R4 Response

Bit		Domain width	Numerical value	Description
47		1	0	Start bit
46		1	0	Transfer bit
[45:40]		6	X	Reserved
[39:8] Parameter field	[39]	1	X	Card is ready
	[38:36]	3	X	Number of I/O functions
	[35]	1	X	Current Memory
	[34:32]	3	X	Filling position
	[31:8]	24	X	I/O ORC
[7:1]		7	X	Reserved
0		1	1	End bit

When an SD I/O card receives command CMD5, the I/O portion of the card is enabled and can respond normally to all subsequent commands. The enable state of the I/O card will remain until the next reset, power down, or receipt of the CMD52 command for I/O reset. Note that a memory-only SD card can respond to a CMD5 command with the correct response: current memory = 1, number of I/O functions = 0. A memory-only SD card designed according to SD memory card specification version 1.0 can detect a CMD5 command as an illegal command and not respond to it. A host that can handle I/O cards will send the CMD5 command, and if the card returns a response R4, the host will determine the card configuration based on the data in the R4 response.

22.2.5.7. R5 (interrupt request)

Only for multimedia cards. Code length = 48 bits. If this response is generated by the host, the RCA field in the parameter is 0x0.

Table 22-26 R5 Response

Bit		Domain width	Numerical value	Description
47		1	0	Start bit
46		1	0	Transfer bit
[45:40]		6	'111111'	CMD40
[39:8] Parameter field	[31:16]	16	X	RCA of successful card or host [31:16]
	[15:0]	16	X	Not defined. Can be used as interrupt data.
[7:1]		7	X	CRC7
0		1	1	End bit

22.2.5.8. R6 (interrupt request)

Only apply to SD I/O cards. This is the normal response of a memory device to a CMD3 command.

Table 22-27 R6 Response

Bit		Domain width	Numerical value	Description
47		1	0	Start bit
46		1	0	Transfer bit
[45:40]		6	'101000'	CMD40
[39:8] Parameter field	[31:16]	16	X	RCA of successful card or host [31:16]
	[15:0]	16	X	Not defined. Can be used as interrupt data.
[7:1]		7	X	CRC7
0		1	1	End bit

When sending a CMD3 command to an I/O-only card, the card's status bits [23:8] will change; at this point, the 16 bits in the response will be the values in the I/O-only SD card: the

- Bit 15 = COM_CRC_ERROR
- Bit 14 = ILLEGAL_COMMAND
- Bit 13 = ERROR
- Bit [12:0] = Reserved

22.2.5.9. R7

Length is 48 bits. The supported voltage information of the card provided by the sending CMD8 response. Bits 16-19 indicate the range of voltages supported by the card. The card uses R7 to feed

back the supportable voltages. In the response, the card will reply with the voltage range and check mode set in the parameters.

Table 22-28 R7 Response

Bit	Domain width	Numerical value	Description
47	1	0	Start bit
46	1	0	Transfer bit
[45:40]	6	'001000'	CMD40
[39:20]	20	'00000h'	Reserved Bit
[19:16]	4	X	Acceptable voltage
[15:8]	8	X	Return of the check mode
[7:1]	7	X	CRC7
0	1	1	End bit

22.3. Software Application Notes

22.3.1. Power Control

Power control can be implemented using the following registers and external circuitry.

Power Enable Register 0x00 - The status of these bits is reflected on the IO pins. Clock Register: The PWRSAV bit of CLKCR allows you to select whether to turn off the clock when the bus is idle. Make sure the "power setting" is correct when power is turned on or off. Normally, the power should be turned off for all cards when the power is cut.

22.3.2. Clock program

The software can select the clock source for the card, choose whether to give the clock and the clock item for dividing the frequency. This function is supported by the register SDIO_CLKCR.

a) CLKEN: enable bit of the clock

b) CLKDIV: Clock dividing factor, which defines the dividing factor of the input clock (SDIOCLK) and the output clock (SDIOCK)

c) SMPCLKSEL: pre-sampling clock edge for CMD and DAT

SMPEN:pre-sampling enable

CKSEL:CMD & DAT output clock selection

These registers are loaded only when the STARTCMD bit and REGSYNC bit of the CMD register are set. When a command is successfully loaded, DWC_mobile_storage clears this bit unless there is already another command in the queue, at which point it gives an HLE (Hardware Locked Error). The software should set the REGSYNC bit (which updates the clock only and does not send commands) and should also set the WAITPEND bit to ensure that the clock parameters do not change during data transfer. Note that even if start_cmd is set to update the clock register, host will not send the command_done signal until the command is completed.

22.3.3. Initialization

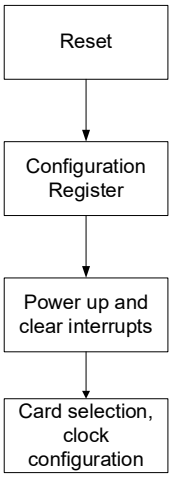


Figure 22-7 Initialization sequence

A reset will initialize the registers, ports, fifo pointers, DMA interface controls, and state machines in the design. After a power-on reset, the software should perform the following actions.

22.3.3.1. Configuration Control Register

Configuration Control Register - For MMC-Ver3.3-only mode, the ODPUEN (bit 24) in the CTRL register is enabled by setting, which is currently set to 1 by default.

22.3.3.2. Electricity on card

Card Power- Before enabling the power supply, make sure that the voltage for the voltage adjustment period is set correctly. Set the corresponding bit of the connected card power supply to 1 by enabling register 0x00.

22.3.3.3. Set interrupt

Set the interrupt mask by clearing the appropriate bit in the interrupt mask register. Set the INTEN bit of the control register. It is recommended that 0xffff_ffff be written to the raw interrupt register to clear any pending interrupts before setting the INTEN bit.

22.3.3.4. Card identification process

The host resets and enters card identification mode, looking for a new card on the bus. In card identification mode, the host resets all cards, verifies the operating voltage range, identifies the cards and asks for the relative card address (RCA) of each card. This operation is done separately on each card's own command signal line, CMD. Only the command signal line (CMD) is used for all data communication in the card identification mode. During card identification, the card should operate at a clock frequency of clock rate FOD (400 kHz).

a. Card reset:

The command GO_IDLE_STATE (CMD0) is a software reset command and sets the MMC and SD memory cards into the Idle State, regardless of the current card state. The reset command (CMD0) is only used for the memory portion of the memory or combo card. To reset only the I/O portion of the I/O card or combo card, use CMD52 to write 1 to the RES bit of the CCCR. Cards in the Inactive State are not affected by this command.

When the host is powered up, all cards are in Idle State, including those that were previously in Inactive State. After power-up or CMD0, the CMD lines of all cards are in input mode, waiting for the

start bit of the next command. These cards are initialized with the card relative address (RCA) and driven with the default 400 kHz clock frequency. The initialization sequence is 80 cycles-this sequence is required to accommodate the increased time after all cards are powered up. To do this, set the AUTOINT (bit 15) of the CMD register.

b. Operating voltage range confirmation

When communication between the host and the card begins, the host may not know the voltage supported by the card, and the card may not know if the voltage provided by the host is supported. Therefore the determination of the operating voltage is required.

The commands defined in the protocol specification include: SEND_OP_COND (CMD1 for MMC), SD_SEND_OP_COND (ACMD41 for SD memory cards), IO_SEND_OP_COND (CMD5 for SDI/O cards), which provide a mechanism for the host to identify and reject cards that do not match the host's required VDD range required by the host. This is accomplished by the host sending the desired VDD voltage window as a parameter to this command. If the card cannot transmit data within the specified range, it must be disconnected from the bus and put into the Inactive State. Otherwise, the card will respond by returning to its VDD range.

If the card can operate at the provided voltage, the response will return the supply voltage and the check mode set in the command parameters. If the card does not work at the provided voltage, it does not return a response and remains in the idle state. Initializing the SDHC card mandatorily sends CMD8 before the ACMD41 command. receiving CMD8 is to let the card know that the host supports the physical layer 2.00 protocol and that the card supports higher versions of the feature.

c. The process of card identification process

For different cards, the card recognition process is different. These cards include MMC, CE-ATA, SD, or SD I/O cards. All types of SD I/O cards are supported, namely SDIO_IO_ONLY card, SDIO_MEM_ONLY card, and SDIO COMBO card. The card identification process steps are as follows:

1. Detects if the card is connected.
2. Identify the card type: SD card, MMC (CE-ATA) or SD I/O card.

   Send CMD5 command. If the host receives a response, it is an SD I/O card; if not, send ACMD41. If the host receives a response, it is an SD card; otherwise, it is an MMC or CE-ATA device.
3. Initialize the card according to the card type.

Use FOD (400 KHz) as the clock source and send the commands in the following command sequence:

- SD card - sends CMD0, ACMD41, CMD2, CMD3;
- SDHC card - sends CMD0, CMD8, ACMD41, CMD2, CMD3;
- SD I/O cards - if the card has no memory port, send CMD52, CMD0, CMD5, CMD3; otherwise, send CMD52, CMD0, CMD5, ACMD41, CMD11 (optional), CMD2, CMD3;

4. Identify the MMC/CE-ATA device

   The CPU should query the 504 bytes of the EXT_CSD register by sending CMD8. If bit 4 is set to 1, the device supports ATA mode; if ATA mode is supported, the CPU should select ATA

mode by setting the (bit 4) ATA bit of byte 191 (CMD_SET) of the EXT_CSD register to activate the use of the ATA command set. The CPU selects the command set using the SWITCH (CMD6) command;

If the CE-ATA device is present, the FAST_IO (CMD39) and RW_MULTIPLE_REGISTER (CMD60) commands will succeed and the data returned will be the CE-ATA reset signal.

d. Set other parameters

According to the specification, the response timeout can be set

ResponseTimeOut = 0x64;

DataTimeOut then selects the following maximum values:

1. (10*((TAAC*Fop)+(100*NSAC));
2. FiFO empty/full read/write delay. The value of de-jittering is 25ms;
3. The threshold value of FIFO is 15 by default.

22.3.4. No data command and no response sequence

To send any non-data commands, the software needs to write CMD registers and CMDARG registers with the appropriate parameters. Using both registers, Host generates the command and will send it to the command bus. host determines the correct transmission by the error bit in the RINTSTS register. When an error or a valid response is received, Host will set the command_done bit in the interrupt register. A short response is placed in response Register0 and a long response is placed in all four response registers. Where Response3 register bit 31 indicates the MSB, Response0 register bit 0 indicates the LSB of the long response.

For basic commands and non-data commands, follow the steps below.

1. Program the command registers with the appropriate command parameters.
2. Program the command registers according to the settings in Table 22-29.
3. Wait for the host to accept the command. Load the command to the accepted execution command and clear the start_cmd bit CMD register, unless there is a command in process, at which point DWC_mobile_storage can load and save the second command in the buffer.

   If DWC_mobile_storage cannot load the command-that is, the command is already in process, the second command is in the buffer, and then wants to proceed to the third command-then a HLE (hardware lock error) is generated .

Table 22-30 No data command register configuration

Parameter	Value	Comment
Default		
STARTCMD	1	
REGSYNC	0	No clock parameter update command
AUTOINIT	0	Can be 1, but only for card reset commands, such as CMD0
ABORTCMD	0	Whether 1 indicates the command to stop data transmission, such as CMD12
AUTOSTOP	0 or 1	0 - No stop command is sent at the end of data

Parameter	Value	Comment
		transmission 1 -Stop command sent at the end of data transmission
RESPECT	0	Command
RESPLEN	0	R2 (long) response can be 1
CMD_INDEX		
WAITPEND	1	0 Send the command immediately 1 Send the command after the previous data transmission is finished
CHECKRESPCRC	1	0 Should not check response CRC 1 Response CRC should be checked

4. Check if there is HLE.
5. Wait for command execution to complete. On receipt of a card response or timeout, DWC_mobile_storage sets the CD bit in the interrupt register. The software can poll this bit or respond to a generated interrupt.
6. Check if response_timeout error, response_CRC error or response error is set. this is possible by polling bits 1,6,8 from RINTSTS, or the corresponding interrupts caused by these errors. If no response error is received, the response is valid. If a response error is received, the software can copy the response from the response register. The software should not modify the clock parameters while the command is being executed.

22.3.5. Data transfer command

The data transfer command transfers data between the memory card and host. To send a data command, host needs a command parameter, total data size, and block size. The software can receive or send data through the FIFO.

Before executing the data transfer command, the software should confirm that the card is not busy and is in the transfer state, which can be done using the CMD13 (select card) and CMD7 (send state) commands, respectively. For the data transfer command, it is important that the same bus width programmed in the card should be set in CLKCR. host generates an interrupt for different conditions during the data transfer, which can be seen in the interrupt register.

1. single or multiple block reads involve the following steps:Write the data size (in bytes) to the DLEN register.
2. Write the block size in bytes to the BLKSIZE register. host expects a block of size BLKSIZE from the card.
3. For cards that have a round-trip delay of more than 0.5 cclk_in cycles, program Card Read Threshold to ensure that the card clock does not stop in the middle of a block of data being transferred from the card to the host; if the Read Threshold feature is not enabled for such a card, then the host system should ensure that the Rx FIFO does not become full of read data during the transfer by ensuring that the Rx FIFO outflow faster than the data pushed into the FIFO; in addition, a large enough FIFO can be provided.

4. Program the SDIO_ARG register with the data address at the beginning of the data read. Use the parameters shown in Table 6-2 to program the command registers. For SD and MMC cards, use CMD17 for single block reads and CMD18 for multi-block reads. For SDIO cards, use CMD53 for both single-block and multi-block transfers. after writing to the CMD register, host starts executing the command; when the command is sent to the bus, a CD interrupt is generated.

Table 22-31 Command register settings for single block and multi-block reads

Parameter	Value	Comment
Default		
STARTCMD	1	
REGSYNC	0	No clock parameter update command
AUTOINIT	0	Can be 1, but only for card reset commands, such as CMD0
ABORTCMD	0	Whether 1 indicates the command to stop data transmission, such as CMD12
AUTOSTOP	0 or 1	0 - No stop command is sent at the end of data transmission 1 - Stop command sent at the end of data transmission
DTMODE	0	Block Transfer
DIR	0	Card Read
DEXPECT	0	Data command
RESPLEN	0	R2 (long) response can be 1
WAITRESP	1	No response to the command can be 0, for example: CMD0,CMD4,CMD15
CMD_INDEX		
WAITPEND	1	0 Send the command immediately 1 Send the command after the previous data transmission is finished
CHECKRESPCRC	1	0 Should not check response CRC 1 Response CRC should be checked

5. The software should look for data error interruptions. If present, the software can terminate the data transfer by sending a stop command.
6. The software should look for Receive_FIFO_Data_request and/or data starvation conditions at the host timeout. In both cases, the software should read data from the FIFO and leave room in the FIFO to receive more data.
7. When the DTO is received, the software should read the remaining data in the FIFO.

22.3.6. Single or multi-block writing

1. Write the data size (in bytes) to the DLEN register.

2. Write the block size in bytes to BLKSIZE register; host sends the block with data size of BLKSIZE. (Is it necessary to send CMD16)
3. program CMDARG register with the data address where the data should be written.
4. FIFO write data; it is usually best to start filling data from the deepest part of the FIFO. Program the command registers using the parameters as shown in Table 6-3. For SD and MMC cards, use CMD24 for single block writes and CMD25 for multi-block writes. For SDIO cards, use CMD53 for both single-block and multi-block transfers.

Table 22-32 Command register settings for single or multi-block writes

Parameter	Value	Comment
Default		
STARTCMD	1	
REGSYNC	0	No clock parameter update command
AUTOINIT	0	Can be 1, but only for card reset commands, such as CMD0
ABORTCMD	0	Whether 1 indicates the command to stop data transmission, such as CMD12
AUTOSTOP	0 or 1	0 - No stop command is sent at the end of data transmission 1 - Stop command sent at the end of data transmission
DTMODE	0	Block Transfer
DIR	1	Card Read
DEXPECT	1	Data command
RESPLEN	0	R2 (long) response can be 1
WAITRESP	1	No response to the command can be 0, for example: CMD0,CMD4,CMD15
CMD_INDEX		
WAITPEND	1	0 Send the command immediately 1 Send the command after the previous data transmission is finished
CHECKRESPCRC	1	0 Should not check response CRC 1 Response CRC should be checked

5. The software should look for data error interruptions. If required, the software can terminate the data transfer by sending the STOP command. 7.
6. the software should look for Transmit_FIFO_Data_request and/or timeout condition starvation from the data. in both cases, the software should write the data to the FIFO.
7. The data command ends when the Data_Transfer_Over interrupt is received. For an open block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is instead of 0, the host should send the stop command after completing the transfer with the given byte count. the completion of the AUTO-STOP command is done through the ACD interrupt of the RINTSTS register.

22.3.7. Data stream reading

Data stream reads are similar to block reads mentioned in single block reads or multi-block reads, except for the following bits in the command register.

Transfer_mode = 1 cmd_index = CMD20; Stream transfer is only allowed per unit bus width.

22.3.8. Data stream writing

Data stream writes are like the block writes mentioned in "Single or multi-block writes", except for the following bits in the command register.

transfer_mode = 1; cmd_index = CMD11;

In a stream transfer, if the byte count is 0, then the software must send the STOP command. If the byte count is not 0, then DWC_mobile_storage sends the STOP command when the given number of bytes completes a transfer. The completion of this AUTO_STOP command is reflected by the Auto_command_done interrupt.

Stream transmission is only allowed per unit bus width.

22.3.9. Send stop and interrupt during transmission

The STOP command terminates the data transfer between the memory card and the DWC_mobile_storage, while the ABORT command only terminates the I/O data transfer between the SDIO_IOONLY and SDIO_COMBO cards.

Send Stop Command - can be sent on the command line when data transfer is in progress; this command can be sent at any time during the data transfer. For information on sending this command, refer to the "No Data Command for No Response Sequence".

You can also use the additional settings of this command to set the command register bits(5-0) to CMD12 and set the 14th bit ABORTCMD to 1. If ABORTCMD is not set to 1, DWC_mobile_storage does not know that the user has stopped data transmission. Reset the command register (WAITPEND) to 0 in order for DWC_mobile_storage to send the command immediately (even if there is a data transfer in progress).

The Send Abort command can only be used on SDIO_IOONLY or SDIO_COMBO cards. To abort a function that is transmitting data, you can use CMD52 to write the function number in the ASx bit (the card's CCCR register, address 0x06). This is a non-data command. For information on sending this command, see No Data Command No Response Sequence.

22.3.10. Suspend or resume sequence

In the SDIO card, data transfer between the I/O function and host can be temporarily stopped using the SUSPEND command; this may be to perform a high priority, data transfer with another function. If needed, data transfer can be resumed using the RESUME command.

The following functions can be implemented by writing the appropriate bits in the CCCR register (Card Common Control Sender) of the SDIO card. To read or write to the CCCR register, use the CMD52 command.

1. SUSPEND data transfer-Non-data transfer command
    a. Check if the SDIO card supports SUSPEND/RESUME protocol; this can be done by the SBS bit in the card's CCCR register @0x08.

b. Check if the data transfer for the desired function is in progress; the currently active function number is reflected on bits 0-3 of the CCCR register.

c. To pause the transfer, set bit 2 of the CCCR register.

d. Poll the clear status of BR (bit 1) and BS (bit 0) of the CCCR. When the currently selected function is using the data bus, the BS (Bus Status) bit is 1; the BR (Bus Release) bit remains 1 until the bus release is completed. When both the BR and BS bits are 0, data transfer for the selected function is suspended.

e. During read data transfer, DWC_mobile_storage can wait for data on the card. If the data transfer is read from the card, then DWC_mobile_storage must be notified after the SUSPEND command completes successfully. d. DWC_mobile_storage then resets the data state machine and exits the wait state. To achieve this, set in the Control register.

f. Wait for data to complete. Get the pending byte to be transferred by reading the TCBCNT register.

2. RESUME data transfer-This is a data command.

a. Check that the card is in the transmit state and confirm that no data is being transmitted from the bus.

b. If the card is in the disconnected state, use CMD7 to select it. The card status can be in response to a CMD52/CMD53 command.

c. Check that the function to be resumed is ready for data transfer; this can be done with the RFx flag of the CCCR. If RF = 1, the function is ready for data transfer.

d. To resume transmission of CMD52, write the function number to the FSx bits (0-3) of the CCCR register @0x0D. the command parameter of CMD52 and write it to CMDARG.

e. Write the block size to the BLKSIZ register; the data will be transferred size in that block.

f. f) Write the number of bytes to the BYTCNT register. This is the total size of the data; that is, the number of bytes remaining to be transferred. It is the software's responsibility to handle the data.

g. Program Command Register; similar to block transfer. See "Multi-block Read" and "Single or Multi-block Write" for details.

h. When the command register is programmed, the command is sent and the function is restored

data transfer. The DF flag (recover data flag) is read. If it is 1, the function has transfer and will start data transfer once the function or memory is restored. If it is 0, then the function has no data to transfer.

i. If the DF flag is 0, then in the case of a read, host waits for data. After the data timeout cycle, it gives a data timeout error.

Table 22-33 CMDARG bit value

CMDARG bits	Content	Value
31	R/W flag	1
30-28	Function Number	0,for CCCR access

27	RAW flag	1,read after write
26		
25-9	Register address	0x0d
8		
7-0	Write data	Function number that is to be resumed

22.3.11. Read Wait Sequence

Read_wait is used only for SDIO cards and can temporarily stop data transfer - from a function or memory - and allow the host to send commands to any function in the SDIO device. The host can delay as long as it needs to.

Operation steps:

1. Check if the card supports the read_wait tool; read SRW (bit 2) of the CCCR register @ 0x08. If this bit is 1, then all functions in the card support the read_wait tool. Use CMD52 to read this bit. 2.
2. If the card supports the read_wait signal, then assert it to register by setting read_wait (bit 6) in CTRL.
3. Clear the read_wait bit in the CTRL register.

22.3.12. Controller/DMA/FIFO Reset Usage

The interaction with the card includes the following.

1. Control of all functions of host.
2. FIFO - Used to store data to be sent or received.
3. DMA - Transfers data between system memory and FIFO if DMA transfer mode is enabled. Reset the controller by setting the controller_reset bit (bit 0) in the CTRL register; this resets the CIU and state machine, and also resets the biu -CIU interface. Because this reset bit is self clearing, after issuing a reset, wait until the bit is cleared.
4. FIFO reset - Resets the FIFO by setting the FIFO reset (bit 1 of the CTRL register); this resets the FIFO pointer and the FIFO counter. Since this reset is self-clearing, wait for this part to be cleared after issuing the reset. The following is the recommended method for issuing a RESET command:
    a. Non-dma transfer mode - Set both controller_reset and fifo_reset; clear the interrupt register by using another write operation to clear any generated interrupts.
    b. Generic DMA mode: set controller_reset, fifo_reset and dma_reset simultaneously; clear the interrupt register by using another write operation to clear any generated interrupts. If DMA needs to be completed, rotate the status register to see if the DMA request is 0 before resetting the DMA interface control and issuing an additional FIFO reset.

22.3.13. Dedicated interrupt

Dedicated interrupt ports do not require special programming. In this case, the application must read all functions connected to the interrupt in order to detect which function generated the interrupt.

22.3.14. Asynchronous interrupt

The asynchronous interrupt mechanism does not require special programming. The application must read all functions connected to the flagged interrupt to detect which function generated the interrupt. In this mode, the card can send interrupts using the DAT[1] line or a dedicated interrupt port.

22.3.15. Error Handling

DWC_mobile_storage implements error checking; ERROR is reflected in the RAWINTS register and can be communicated to software via interrupts, or software can poll these bits. After power-up, interrupts are disabled (int_enable in the CTRL register is 0) and all interrupts are masked (bits 0-31 of the INTMASK register; default is 0)

Error handling :

a. Response and data timeout errors

   For response timeouts, the software can retransmit the command. dwc_mobile_storage has not received the data start bit at the data timeout, either for the first block or for an intermediate block - within the timeout time, so the software can retry the entire data transfer again or start the transfer from the specified block. Later, by reading the contents of the TCBCNT, you can find out how many bytes the software can still have to copy.

b. Response error

   Set when an error is received during response reception. For example, the response appears in the response register as invalid. The software can resend the command.

c. Data error

   Set when a data reception error is observed; for example, data CRC, start bit not found, end bit not found, etc. These errors can occur in any data block - the first block, the middle block, or the last block. Upon receipt of an error, the software can issue a STOP or ABORT command and resend the command for the entire data or part of it.

d. Hardware lock error

   When DWC_mobile_storage cannot set software to load a command. DWC_mobile_storage attempts to load a command when software sets the STARTCMD bit in the CMD register. This error is raised if the command buffer is already filled with commands. The software must reload the command.

e. FIFO underrun/overrun error

   If the FIFO is full, the software tries to write data in the FIFO and then sets an overrun error. Conversely, if the FIFO is empty and the software attempts to read from the FIFO, an underrun error is set. The software should read the fifo_empty or fifo_full bit in the status register before reading or writing data from the FIFO.

f. Data scarcity due to host timeout

   Data scarcity is caused when DWC_mobile_storage is waiting for software intervention to transfer data to or from the FIFO, but the software does not do so within the specified timeout period. In this case, while a read transfer is in progress, the software reads data from the FIFO and provides space for further data reception. While a transfer operation is in progress, the software should fill the FIFO with data in order to start transferring data to the card.

g. CRC errors in commands

If a command CRC error is detected, the CE-ATA device does not send a response and the DWC_mobile_storage response is expected to time out. an error occurs in the ATA layer identifying the MMC transport layer.

h. Write Operations

Any known MMC transport layer error for the device results in the termination of an unexecuted ATA command. the ERR bit is in the ATA status register and the corresponding error code is sent to the ATA error register. If nIEN=0, CCS is sent to the host. if device interrupts are not enabled (nIEN=1), the device completes the entire Data Unit Count if the host controller does not terminate the ongoing transfer.

i. Read Operation

The host controller may send a Command Completion Signal Disable Signal (CCSD), after a Stop Transfer command is sent (CMD12), to abort a read transfer.

If the host controller detects an MMC transport layer error, the host completes the ATA command with an error status. The host controller can issue a Command Completion Signal Disable (CCSD) followed by a Stop Transfer (CMD12) to abort the read transfer. The host can also transfer the entire Data Unit Count byte without interrupting the data transfer.

22.3.16. CT_ATA Operation

Introduces the CE-ATA data transfer command. Information on basic settings and interrupts generated under different conditions can be found in the Design Transfer command

22.3.16.1. Reset and device recovery

Before starting the CE-ATA operation, the host should perform the MMC reset and initialization process. After completing the normal MMC reset and initialization process, the host should query the initial ATA value using RW_REG/CMD39. By default, the MMC block size is 512 bytes, indicated by bits 1:0 of the srcControl register inside the CE-ATA device. The host can negotiate to use either a 1KB or 4KB MMC block size. The device indicates the MMC block size it can support through the srcCapabilities register; the host reads this register to negotiate the MMC block size. This is done when the host controller writes the MMC block size to the device's srcControl register with bits 1:0.

22.3.16.2. ATA Task File Transfer

The host software stack writes the task file image to the FIFO before setting the CMDARG and CMD registers.The host processor then sets the address and byte count in the CMDARG before setting the CMD (offset 0x2C) register bits. For RW_REG, the CE-ATA device has no command completion signal.

22.3.16.3. ATA task file transfer using RW_MULTIPLE_REGISTER

This command involves the transfer of data between the CE-ATA device and host. To send the data command, host needs a command parameter, total data size and block size. The software can receive or send data via FIFO ATA file transfer (read or write) involves the following steps.

1. write the data size in bytes to the BYTCNT register
2. Write the block size in bytes to the BLKSIZ register

3. DWC_mobile_storage requires a separate block transfer.
4. Write CMDARG register with starting register address.

Table 22-34 CMDARG Programming Register

CMDARG bits	Content	Value
31	R/W flag	1or0
30-24	0	Reserved ,bits cleared to 0 by host processor
23:18	0	Starting register address for read/write; Dword aligned
17:16	0	Register address; Dword aligned
15:8	0	Reserved; bits cleared to 0 by host processor
7:2	16	Number of bytes to read/write; integral number of Dwords
1:0	0	Byte count in integral number of Dword

Table 22-35 CMD Register

Parameter	Value	Comment
Default		
STARTCMD	1	
REGSYNC	0	No clock parameter update command
DIR	1 or 0	Read the card, or write the card
IEN	0	Command Completion signals Done signals
ATACMD	1 or 0	CE_ATA CMD
DEXPECT	1	Data order expectations
RESPLEN	0	R2 (long) response can be 1
WAITRESP	1	No response to the command can be 0, for example: CMD0,CMD4,CMD15
CMD_INDEX		
WAITPEND	1	0 Send the command immediately 1 Send the command after the previous data transmission is finished
CHECKRESPCRC	1	0 Should not check response CRC 1 Response CRC should be checked

22.3.16.4. ATA transfer using RW_MULTIPLE_REGISTER

While waiting for RW_BLK of the Command Completion Signal (CCS), the host can send a Command Completion Signal Disable (CCSD). Send CCSD DWC_mobile_storage sends CCSD to the CE-ATA device if the send_ccsd bit is set in the CTRL register; this bit should only be set after receiving a response from RW_BLK. After sending CCSD mode, send the internally generated Stop (CMD12) command. If the send_auto_stop_ccsd bit is also set when the controller is programmed to send CCSD mode, DWC_mobile_storage sends an internally

generated STOP command on the CMD line. After the STOP command is sent, DWC_mobile_storage sets the Auto command Done bit in the RINTSTS register.

22.4. Register Configuration

22.4.1. SDIO power control register (SDIO_POWER)

Address Offset: 0x00

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															PWRCTRL
															rw

Bit	Name	R/W	Reset Value	Function
31: 1	Reserved	RW	0	Reserved, always read as 0.
0	PWRCTRL	RW	0	PWRCTRL: Power supply control bits These bits are used to define the current functional state of the card clock: 0: Power is off and the card clock is stopped. 1: Reserved power-up state.

Note: This register cannot be written for 7 HCLK clock cycles after writing data.

22.4.2. SDIO clock control register (SDIO_CLKCR)

Address Offset: 0x04

Reset value: 0x0000 7000

The SDIO_CLKCR register controls the SDIO_CK output clock.

31	30	29	28	27	26	25	24	23	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	CKSEL	SMPEN	SMP-CLKSEL	WIDBUS		PWRSAV	CLKEN	CLKDIV[0:7]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31：15	Reserved	RW	0	Reserved, always read as 0.
14	CKSEL	RW	1	CMD & DAT output clock selection: 0: SD_CLK clock offset by 90 degrees output; 1: SD_CLK clock offset 180 degrees output;
13	SMPEN	RW	1	Pre-sampling enable:

Bit	Name	R/W	Reset Value	Function
				0: no pre-sampling clock is used; 1: pre-sampling clock is used;
12	SMPCLKSEL	RW	1	CMD & DAT pre-sampling clock selection (both rising edge): 1: SD_CLK clock; 0: SD_CLK clock offset by 270 degrees;
11：10	WIDBUS	RW	0	WIDBUS: Wide bus mode enable bit 00: Default bus mode, use SDIO_D0. 01: 4-bit bus mode, use SDIO_D[3:0]. 10/11: 8-bit bus mode, use SDIO_D[7:0].
9	PWRSAV	RW	0	PWRSAV: Power saving configuration bit. To save power, set the PWRSAV bit to turn off the SDIO_CK clock output when the bus is idle. 0: Always output SDIO_CK. 1: SDIO_CK is output only when there is bus activity.
8	CLKEN	RW	0	CLKEN: Clock enable bit 0: SDIO_CK is off. 1: SDIO_CK enable.
7:0	CLKDIV	RW	0	CLKDIV: Clock divide factor . This field defines the divide factor between the input clock (SDIOCLK) and the output clock (SDIO_CK): SDIO_CK frequency = SDIOCLK/[CLKDIV*2]

Note:

1. When the SD/SDIO card or multimedia card is in recognition mode, the frequency of SDIO_CK must be below 400 kHz.
2. When all cards are given the corresponding addresses, the clock frequency can be changed to the maximum frequency allowed by the card bus.
3. This register cannot be written within 7 HCLK clock cycles after writing data. For SD I/O cards, SDIO_CK can be stopped during read wait, when the SDIO_CLKCR register does not control SDIO_CK.

22.4.3. SDIO parameter register (SDIO_ARG)

Address Offset: 0x08

Reset value: 0x0000 0000

The SDIO_ARG register contains the 32-bit command parameter that will be sent to the card as part of the command. the SDIO_CLKCR register controls the SDIO_CK output clock.

31:0
CMDARG[31:0]
RW

Bit	Name	R/W	Reset Value	Function

31：0	CMDARG	RW	0	CMDARG: Command argument The command argument is part of the command sent to the card. If a command contains an argument, this register must be loaded before the command can be written to the command register.

22.4.4. SDIO command register (SDIO_CMD)

Address Offset: 0x0C

Reset value: 0x2000 0000

The SDIO_CMD register contains the command index and command type bits. The command index is sent to the card as part of the command. The Command Type bit controls the Command Channel State Machine (CPSM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STARTCMD	Res	USEHOLDREG	Res	BOOTMODE	BOOTDIS	BOOTACK	BOOTEN	IEN	ATACMD	REGSYNC	Res				
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**
AUTOINIT	ABORTCMD	WAITPEND	AUTOSTOP	DTMODE	DIR	DEXPECT	CHECKRESPCRC	RESPLEN	WAITRESP	rwCMDINDEX[0:5]					
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw					

Bit	Name	R/W	Reset Value	Function
31	STARTCMD	RW	0	Start command The start command. Once the command is received by the CIU, the bit is cleared. When this bit is set, the host is not able to write to the command register. If a write is attempted, a hardware lock error will be set in the interrupt register. Once the command is sent and the response is received from the SD_MMC_CEATA card, the command completion bit is set in the interrupt register.
30	RESERVED	RW	0	RESERVED
29	USEHOLDREG		1	Forced to 1, software reads 0
28	RESERVED	RW	0	RESERVED
27	BOOTMODE	RW	0	Boot Mode 0 -Forced boot operation 1 - Standby boot operation
26	BOOTDIS	RW	0	Disable_boot Disable boot. When this bit of the software is set together with start_cmd, the CIU will terminate the

Bit	Name	R/W	Reset Value	Function
				boot operation. "disable_boot" and "enable_boot" cannot be set at the same time.
25	BOOTACK	RW	0	Expect_boot_ack Expected boot response. When Software sets this bit together with enable_boot. When the mode on the selected card is 0-1-0 and CIU expect confirms boot.
24	BOOTEN	RW	0	Enable_boot Enable Boot-This bit should only be set to force boot mode. When Software sets this bit together with start_cmd, the CIU pulls down the boot line of the corresponding card by asserting CMD boot. "disable_boot" and "enable_boot" cannot be set at the same time.
23	IEN	RW	0	CCS_expected 0 - No interrupts are enabled in the CE-ATA device (nIEN = 1 in the ATA control register), or the command receives an expect from the device CCS. 1 - Interrupts are enabled on the CE-ATA device (nIEN = 0), and the RW_BLK command receives a command completion signal from the CE-ATA device.
22	ATACMD	RW	0	ATACMD 0 The host is not performing read access (RW_REG or RW_BL) to the CE-ATA device. 1 The host performs a read access (RW_REG or RW_BLK) to the CE-ATA device.
21	REGSYNC	RW	0	Update_clock_registers_only 0 - normal command sequence 1 - No command is sent, just update the clock register values to the card clock domain Transfer the following register values to the card clock field: CLKDIV, CLKENA. Change the card clock (change frequency, cut or turn on, set low frequency mode); provides the ability to change the clock frequency or stop the clock without sending a command to the card During the normal command sequence, update_clock_registers_only = 0, the following control registers are transferred from the BIU to the CIU: CMD, CMDARG, TMOUT, BLKSIZ, bytct. The CIU uses the new register values for the card's new command sequence.

Bit	Name	R/W	Reset Value	Function
				When the bit is set, there is no command completion interrupt because no commands are sent to the SD_MMC_CEATA card.
20:16	Reserved	RW	0	
15	AUTOINIT	RW	0	Send_initialization 0 - no initialization sequence is sent before sending this command (80 clocks) 1 -Send the initialization sequence before sending this command
14	ABORTCMD	RW	0	Stop_abort_cmd 0 - Neither stop nor abort command stops the data transfer currently in progress. If abort is sent to the currently selected function number or is not in data transfer mode, then bit should be set to 0. 1 - The stop or abort command stops the data transfer currently in progress in order to stop it.
13	WAITPEND	RW	0	Wait_prvdata_complete 0 - send the command immediately, even if the previous data transfer has not yet completed 1 - Wait for the previous data transfer to complete before sending the command. wait_prvdata_complete = 0 is usually used to query the card status while data is being transferred or to stop the current data transfer; Card_number should be the same as the previous command.
12	AUTOSTOP	RW	0	Send_auto_stop 0 - No stop command is sent at the end of data transmission 1 -Send stop command at the end of data transmission Note: The send_auto_stop bit is loaded in the command path when it is set in the command register. auto-stop command helps to send the exact number of data bytes, stream read or write using MMC, and multi-block read or write for SD card.
11	DTMODE	RW	0	Transfer_mode 0 - Block data transfer command 1 -Stream data transfer command Does not care if it is the expected data or not.

Bit	Name	R/W	Reset Value	Function
10	DIR	RW	0	Read/Write 0 - read from the card 1 - write to the card Does not care whether data is expected in the card or not.
9	DEXPECT	RW	0	Data_expected 0 - no data transfer (read/write) expected 1 -expected data transfer (read/write)
8	CHECKRESPCRC	RW	0	Check_response_crc 0 -No check response CRC 1 - check response CRC Some command responses do not return a valid CRC bit. The software should disable CRC checking for these commands to disable CRC checking for the controller.
7	RESPLEN	RW	0	Response_length 0 - Expect a short response from the card 1 - Expect a long response from the card
6	WAITRESP	RW	0	Response_expect 0 -No response expected from the card 1 - Expect to receive a response from the card
5:0	CMDINDEX	RW	0	CMDINDEX Command Index

Note:

1. This register cannot be written within 7 HCLK clock cycles after writing data.
2. Multimedia card can send 2 types of response: 48-bit long short response, or 136-bit long response. sd card and sd i/o card can only send short response, the parameter can be changed according to the type of response, the software will distinguish the type of response according to the command sent. ce-ata device only sends short response.

22.4.5. SDIO command response register (SDIO_RESPCMD)

Address Offset: 0x10

Reset value: 0x0000 0000

The SDIO_RESPCMD register contains the command index from the last received command response. If the transmitted command response does not contain a command index (long response or OCR response), the content of the RESPCMD field is unknown, although it should contain 1111111b (the value of the reserved field in the response).

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

Bit	Name	R/W	Reset Value	Function
31：6	Reserved	RW	0	Reserved, always read as 0.
5:0	RESPCMD	R	0	RESPCMD: Response command index. Read-only bit that contains the command index of the last received command response.

22.4.6. SDIO command response 1..4 register (SDIO_RESPX)

Address Offset: 0x14+4*(x-1), where x = 1...4

Reset value: 0x0000 0000

The SDIO_RESP1/2/3/4 registers contain the status of the card, i.e. part of the information about the response received.

31:0
RESPX[31:0]
r

Bit	Name	R/W	Reset Value	Function
31：0	CARDSTATUSX	R	0	CARDSTATUSx: See the table below

Depending on the response status, the card status length is 32 bits or 127 bits.

Table 22-36 Response type and SDIO_RESPx register

Register	Short response	Long Response
SDIO_RESP1	Card Status [31:0]	Card Status [127:96]
SDIO_RESP2	No need to	Card Status [95:64]
SDIO_RESP3	No need to	Card Status [63:32]
SDIO_RESP4	No need to	Card Status [31:1]

The highest bit of the card status is always received first, and the lowest bit of the SDIO_RESP3 register is always 0.

22.4.7. SDIO data timer register (SDIO_TMOUT)

Address Offset: 0x24

Reset value: 0XFFFFFF40

The SDIO_DTIMER register contains the data timeout in units of card bus clock cycles.

A counter loads the value from the SDIO_DTIMER register and decrements the count when the Data Channel State Machine (DPSM) enters the Wait_R or Busy states, and sets the timeout flag if the counter decrements to 0 when the DPSM is in these states.

31:8	7:0
DATATIME[31:8]	RESPTIME[7:0]
RW	RW

Bit	Name	R/W	Reset Value	Function
31 : 8	DATATIME	RW	0xFFFFFF	DATATIME: Data timeout period . The data timeout period in terms of card bus clock cycles.
7:0	RESPTIME	RW	0X40	RESPTIME: Command timeout period . Data timeout period in card bus clock cycles.

Note: The Data Timer Register and Data Length Register must be written before the Data Control Register can be written for data transfer.

22.4.8. SDIO data block length register (SDIO_BLKSIZE)

Address Offset: 0x28

Reset value: 0x0000 0200

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

Bit	Name	R/W	Reset Value	Function
31: 16		Res	0	Reserved always 0
15:0	DBLOCKSIZE	RW	0x200	DBLOCKSIZE: Data block size When block data transfer mode is selected, this field defines the data block length

22.4.9. SDIO Data Length Register (SDIO_DLEN)

Address Offset: 0x2C

Reset value: 0x0000 0200

The SDIO_DLEN register contains the length of the data bytes to be transferred. This value is loaded into the data counter when the data transfer starts.

31:25	24:0
Res	DATALENGTH
	RW

Bit	Name	R/W	Reset Value	Function
31:25		Res	0	Reserved always 0
24:0	DATALENGTH	RW	0x200	DATALENGTH: Data length value The number of data bytes to be transferred.

Note: For block data transfer, the value in the Data Length register must be a multiple of the data block length (SDIO_BLKSIZ). The Data Timer Register and Data Length Register must be written before writing to the Data Control Register for data transfer.

22.4.10. SDIO control register (SDIO_CTRL)

Address Offset: 0x30

Reset value: 0x01000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							ODPUEN	Res							
							RW								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				CEATAINTEN	AUTOSTOPCCSD	CCSDEN	ABORTRD	AUTOIRQRESP	READWAIT	DMAEN	INTEN	Res		FIFORST	SDIORST
				RW	RW	RW	RW	RW	RW	RW	RW			RW	RW

Bit	Name	R/W	Reset Value	Function
31：25	Reserved			
24	ODPUEN	RW	1	Enable_od_pullup External OD pullup: 0: Disable 1: Enable CMD pull-up on line, when pull-up enable SDIO output 0 or high resistance, will not output 1; (used during MMC initialization, need to modify after the end of identification, can be modified by configuring gpio)
23：12	Reserved			
11	CEATAINTEN	RW	0	Ceata_device_interrupt_status 0: In CE-ATA devices, interrupts are not enabled; 1: Interrupt enabled in the CE-ATA device;
10	AUTOSTOPCCSD	RW	0	Send_auto_stop_ccsd 1: Send STOP automatically after sending CCSD to CE-ATA; After sending, this bit is automatically cleared; controller_reset can reset this BIT, or software can write 0
9	CCSDEN	RW	0	Send_ccsd 1: Sending CCSD to CE-ATA; When set, DWC_mobile_storage sends CCSD to the CE-ATA device.

Bit	Name	R/W	Reset Value	Function
				This bit is set by software only if the current command expects CCS (i.e., RW_BLK) and interrupts are enabled in the CE-ATA device. Once the CCSD mode is sent to the device, SDIO automatically clears the send_ccsd bit. It also sets the command complete (CD) bit in the RINTSTS register and generates an interrupt to the host if the command complete interrupt is not masked.
8	ABORTRD	RW	0	Abort_read_data 0 - No change 1 - When the hang command is issued during a read transfer, the software polls the card to find out when the hang occurred. Once a hang occurs, the software sets a bit to reset the data state machine and waits for the next block of data. The bit is automatically cleared once the data state machine is reset to idle. Used for SDIO card pending queue.
7	AUTOIRQRESP	RW	0	Send_irq_response 0 - no change 1 -Send automatic IRQ response Once the response is sent, the bit is automatically cleared. In order to wait for an MMC card interrupt, the host sends CMD40, and SDIO waits for an interrupt response from the MMC card. Also, if the host wants the SDIO to exit the wait for interrupt state, this bit can be set, at which point the SDIO command state machine sends a CMD40 response on the bus and returns to the idle state.
6	READWAIT	RW	0	Read_wait 0 - Clear read wait 1 - Assert read wait Used to send read waits to the SDIO card.
5	DMAEN	RW	0	DMAEN 0 - Disable DMA transfer mode 1 - Enables DMA transfer mode
4	INTEN	RW	0	Int_enable Global interrupt enable/disable bit. 0 - Disable interrupts 1 - Enables interrupts The int port is 1 only when this bit is 1 and one or more unmasked interrupt bits are set

Bit	Name	R/W	Reset Value	Function
3 : 2	Reserved			
1	FIFORST	RW	0	FIFO_reset 0 - No change 1 - Reset data FIFO, reset FIFO pointer To reset the FIFO, the firmware should set the bit to 1. This bit will be cleared automatically after the reset operation is completed
0	SDIORST	RW	0	Controller_reset 0 - No change 1 - Reset controller To reset the controller, the firmware should set the bit to 1. This bit is automatically cleared.

22.4.11. SDIO Status Register (SDIO_STA)

Address Offset: 0x34

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		FIFOCNT													Res
		R													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res						CARDBSY	CARDPRESENT	CMDFSM				FIFOF	FIFOE	TXWMARK	RXWMARK
						R	R					R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:30	RESERVED			
29:17	FIFOCNT	R	0	FIFO count
16:10	RESERVED			
9	CARDBSY	R	0	Data_busy Check the opposite of the card_data[0] state 0 - Card data is not busy 1 - Card data busy
8	CARDPRESENT	R	0	Data_3_status Select card_data [3]; check if the card exists 0 - card does not exist 1 - card exists

Bit	Name	R/W	Reset Value	Function
7:4	CMDFSM	R	0	Cmd_fsm_states Command FSM status. 0 - Idle 1 - Send init sequence 2 - Tx cmd start bit 3 - Tx cmd Tx bits 4 - Tx cmd index + parameters 5 - Tx cmd crc7 6 - Tx cmd end bit 7 - Rx response start bit 8 - Rx response IRQ response 9 - Rx response tx bit 10 - Rx response cmd index 11 - Rx response data 12 - Rx response crc7 13 - Rx response end bit 14 - Cmd path wait for NCC 15 - Wait; CMD-to-response turnaround
3	FIFOF	R	0	FIFO_full FIFO full
2	FIFOE	R	1	FIFO_empty FIFO empty
1	TXWMARK	R	1	FIFO_tx_watermark The FIFO reached the transmission watermark; it does not meet the conditions for data transmission.
0	RXWMARK	R	0	FIFO_rx_watermark The FIFO reached the output water level line; not eligible for data transfer

22.4.12. SDIO interrupt status register (SDIO_INTSTS)

Address Offset: 0x38

Reset value: 0x0000 0000

In the corresponding position '1', this bit of SDIO_INTSTS is cleared to 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															SDIOINT
															Rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EBE	ACD	SBE	HLE	FRUN	HTO	DRTO_BDS	RTO_BAR	DCRC	RCRC	RXDR	TXDR	DTO	CD	RE	CAD
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

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16	SDIOINT	Rc_w1	0	Sdio_int_mask 0 - No SDIO interrupt from the card 1 - SDIO interrupt from card
15	EBE	Rc_w1	0	Bit 15 - End-bit error (read)/write no CRC (EBE)
14	ACD	Rc_w1	0	Bit 14 - Automatic Command Completion (ACD)
13	SBE	Rc_w1	0	Bit 13 -Start Bit Error (SBE)
12	HLE	Rc_w1	0	Bit 12 - Hardware Lockout Write Error (HLE)
11	FRUN	Rc_w1	0	Bit 11 - FIFO underrun/overflow error (FRUN)
10	HTO	Rc_w1	0	Bit 10 - host timeout (HTO)
9	DRTO_BDS	Rc_w1	0	Bit 9 - Data Read Timeout (DRTO_BDS)
8	RTO_BAR	Rc_w1	0	Bit 8 - Response Timeout (RTO_BAR)
7	DCRC	Rc_w1	0	Bit 7 - Data CRC error (DCRC)
6	RCRC	Rc_w1	0	Bit 6 -Response CRC error (RCRC)
5	RXDR	Rc_w1	0	Bit 5 -Receive FIFO data request (RXDR)
4	TXDR	Rc_w1	0	Bit 4 - Send FIFO data request (TXDR)
3	DTO	Rc_w1	0	Bit 3 - Data Transfer (DTO)
2	CD	Rc_w1	0	Bit 2 - Command Completion (CD)
1	RE	Rc_w1	0	Bit 1 -Response Error (RE)
0	CAD	Rc_w1	0	Bit 0 - Card Detection (CAD)

22.4.13. SDIO interrupt mask register (SDIO_INTMASK)

Address Offset: 0x3C

Reset value: 0x0000 0000

In the corresponding position '1', the SDIO_MASK interrupt mask register determines which status bit generates the interrupt.

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

EBEIE	ACDIE	SBEIE	HLE	FRUNIE	HTOIE	DRTO_BDSIE	RTO_BARIE	DCRCIE	RCRCIE	RXDRIE	TXDRIE	DTOIE	CDIE	REIE	CADIE
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:17	Reserved			
16	SDIOINTIE	RW	0	Sdio_int_mask 0 - No SDIO interrupt from the card 1 - SDIO interrupt from card
15	EBEIE	RW	0	Bit 15 - End-bit error (read)/write no CRC (EBE)
14	ACDIE	RW	0	Bit 14 - Automatic Command Completion (ACD)
13	SBEIE	RW	0	Bit 13 - Start Bit Error (SBE)
12	HLEIE	RW	0	Bit 12 - Hardware Lockout Write Error (HLE)
11	FRUNIE	RW	0	Bit 11 - FIFO underrun/overflow error (FRUN)
10	HTOIE	RW	0	Bit 10 - host timeout (HTO)
9	DRTO_BDSIE	RW	0	Bit 9 - Data Read Timeout (DRTO_BDS)
8	RTO_BARIE	RW	0	Bit 8 - Response Timeout (RTO_BAR)
7	DCRCIE	RW	0	Bit 7 - Data CRC error (DCRC)
6	RCRCIE	RW	0	Bit 6 -Response CRC error (RCRC)
5	RXDRIE	RW	0	Bit 5 -Receive FIFO data request (RXDR)
4	TXDRIE	RW	0	Bit 4 - Send FIFO data request (TXDR)
3	DTOIE	RW	0	Bit 3 - Data Transfer (DTO)
2	CDIE	RW	0	Bit 2 - Command Completion (CD)
1	REIE	RW	0	Bit 1 -Response Error (RE)
0	CADIE	RW	0	Bit 0 - Card Detection (CAD)

22.4.14. SDIO FIFO threshold register (SDIO_FIFOTH)

Address Offset: 0x40

Reset value: 0x000F 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				RXWMARK											
				RW											
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res				TXWMARK											
				RW											

Bit	Name	R/W	Reset Value	Function
31:28	Reserved			
27:16	RXWMARK	RW	F	The FIFO threshold water mark when the card receives data. When the FIFO data count exceeds this number, a DMA/FIFO request is made. When the packet ends, the request is generated to complete the remaining data, regardless of the threshold programming. In non-DMA mode, when Receive FIFO Threshold Interrupt (RXDR) is enabled, an interrupt is generated instead of a DMA request. During packet completion, if the threshold is programmed to be greater than any remaining data, no interrupt is generated. It is the host's responsibility to read the remaining bytes before seeing a data transfer interrupt. In DMA mode, at the end of the packet, even if the remaining bytes are less than the threshold, the DMA request makes a single transfer to clear the remaining bytes before setting the "Data Transfer Complete" interrupt.
15:12	Reserved			
11:0	TXWMARK	RW	F	When data is transferred to the card, the FIFO threshold water mark. When the FIFO data count is less than or equal to this number, a DMA/FIFO request is made. If interrupts are enabled, an interrupt occurs. At the end of the packet, the request or interrupt is generated, regardless of the threshold programming. In non-DMA mode, when the transmit FIFO threshold (TXDR) interrupt is enabled, an interrupt is generated instead of a DMA request. At the end of the packet, at the last interrupt, the host is responsible for filling the FIFO with only the remaining bytes needed (not before the FIFO is full, and not after the CIU finishes data transfer, because the FIFO may not be empty). In DMA mode, at the end of the packet, if the last transfer is smaller than the burst size, the DMA controller does a single cycle until the required bytes are transferred.

22.4.15. SDIO send to card counter (SDIO_TCBCNT)

Address Offset: 0x44

Reset value: 0x0000 0000

31:0
TRANSCARDBYTECOUNT
R

Bit	Name	R/W	Reset Value	Function
31:0	TRANSCARDBYTECOUNT	R	0	The number of bytes transferred to the card by the CIU unit.

22.4.16. SDIO send to FIFO counter (SDIO_TBBCNT)

Address Offset: 0x48

Reset value: 0x0000 0000

31:0
trans_fifo_byte_count
R

Bit	Name	R/W	Reset Value	Function
31:0	trans_fifo_byte_count	R	0	The number of bytes transferred between HOST/DMA memory and BIU FIFO.

22.4.17. SDIO data FIFO register (SDIO_FIFODATA)

Address Offset: 0x200

Reset value: 0x0000 0000

The receive and transmit FIFOs are 32-bit wide read or write sets of registers, which contain 32 registers on 32 consecutive addresses. Note that the data FIFO registers include send and receive FIFOs, so it is important to divide these 32 addresses into a group of 16, with send and receive each taking half. And in each time we read and write, the most is to read the receive FIFO or write the general size of the send FIFO data.

The CPU can use the FIFO to read and write multiple operands.

31:0
FIFODATA
RW

Bit	Name	R/W	Reset Value	Function
31: 0	FIFODATA	RW	0	FIFODATA: Receive and transmit FIFO data The FIFO host does not support simultaneous reading and writing from one port. For debugging purposes, the software may try to write to the FIFO and read back the data with uncertain results, as the design does not support read/write access from the same port.

22.4.18. SDIO register map

0x000	SDIO_POWER	Reserved																														PWRCTRL
	Read/Write																															rw
	Reset Value	0																														0
0x004	SDIO_CLKCR	Reserved																						CKSEL	SMPEN	SMPCLKSE	WIDBUS	PWRSAV	CLKEN	CLKDIV		
	Read/Write																							rw	rw	rw	rw	rw	rw	rw		
	Reset Value	0																						1	1	1	0	0	0	0		
0x008	SDIO_ARG	CMDARG																														
	Read/Write	rw																														
	Reset Value	0																														
0x00C	SDIO_CMD	STARTCMD	Reserved	USEHOLDREG	RESERVED	BOOTMODE	BOOTDIS	BOOTACK	BOOTEN	IEN	ATACMD	REGSYNC		AUTIINIT	ABORTCMD	WAITPEND	AUTOSTOP	DTMOODE	DIR	DEXPECT	CHECKRESPCRC	RESPLEN	WAITRESP	CMDINDEX								
	Read/Write	rw				rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw								
	Reset Value	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x010	SDIO_RESPCMD	Reserved																													RESPCMD	
	Read/Write																														r	
	Reset	0																													0	

	Value		
0x14	SDIO_RESP0	SDIO_RESP1	
	Read/Write	r	
	Reset Value	0	
0x18	SDIO_RESP1	SDIO_RESP2	
	Read/Write	r	
	Reset Value	0	
0x1C	SDIO_RESP2	SDIO_RESP3	
	Read/Write	r	
	Reset Value	0	
0x20	SDIO_RESP3	SDIO_RESP4	
	Read/Write	r	
	Reset Value	0	
0x24	SDIO_DTIMER	DATATIME	RESPTIME
	Read/Write	rw	rw
	Reset Value	0xFFFFFF	0x40

Offset	Register		
0x28	SDIO_BLKSIZ	Reserved	DBLOCKSIZE
	Read/Write		rw
	Reset Value	0	0x200

Offset	Register	
0x2C	SDIO_DLEN	DATALENGTH
	Read/Write	rw
	Reset Value	0x200

Offset	Register														
0x30	SDIO_CTRL	Reserved	ODPUEN	Reserved	CEATAINTEN	AUTOSTOPCCSD	CCSDEN	ABORTRD	AUTOIRQRESP	READWAIT	DMAEN	INTEN	Reserved	FIFORST	SDIORST
	Read/Write		rw		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
	Reset Value	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register											
0x34	SDIO_STATUS	Reserved		FIFOCNT	Reserved	CARDBSY	CARDPRESENT	CMDFSM	FIFOF	FIFOE	TXWMARK	RXWMARK
	Read/Write			r		r	r	r	r	r	r	r
	Reset Value	0	0	0	0	0	0	0	0	1	1	0

Offset	Register																		
0x38	SDIO_INTSTS	Reserved	SDIOINT	EBE	ACD	SBE	HLE	FRUN	HTO	DRTO_BDS	RTO_BAR	DCRC	RCRC	RXDR	TXDR	DTO	CD	RE	CAD

	Read/Write		rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1	rcw1
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	SDIO_INTMASK	Reserved	SDIOINTIE	EBEIE	ACDIE	SBEIE	HLEIE	FRUNIE	HTOIE	DRTOIE	RTOIE	DCRCIE	RCRCIE	RXDRIE	TXDRIE	DTOIE	CDIE	REIE	CADIE		
	Read/Write		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x40	SDIO_FIFOTH	Reserved	RXWMARK							Reserved	TXWMARK										
	Read/Write		rw								rw										
	Reset Value	0	f							0	f										
0x44	SDIO_TCBCNT	TCBCNT																			
	Read/Write	r																			
	Reset Value	0																			
0x48	SDIO_TBBCNT	TBBCNT																			
	Read/Write	r																			
	Reset Value	0																			

0x200	SDIO_FIFO	FIFODATA
	Read/Write	rw
	Reset Value	0x00

23. USB Full Speed Device Interface (USB)

23.1. Introduction

The USBD peripheral implements the interface between the USB2.0 full-speed bus and the APB1 bus.

The USBD peripheral supports USB suspend/wake operations and can stop the device clock for low power consumption.

23.2. Main features

- Compliant with USB 2.0 full-speed device specifications
- Configurable with 1 to 8 USB endpoints (ENDPOINT0 ~ ENDPOINT7)
- Uses a dedicated 512-byte data memory
- CRC (cyclic redundancy check) generation/checking, reverse non-return-to-zero (NRZI) encoding/decoding and bit padding
- Supports control transfer/synchronous transfer/bulk transfer/interrupt transfer
- Support double buffer mechanism for bulk synchronous endpoints
- Support USB suspend/wake up operation
- Frame-locked clock pulse generation

23.3. Function Description

23.3.1. Module Block Diagram

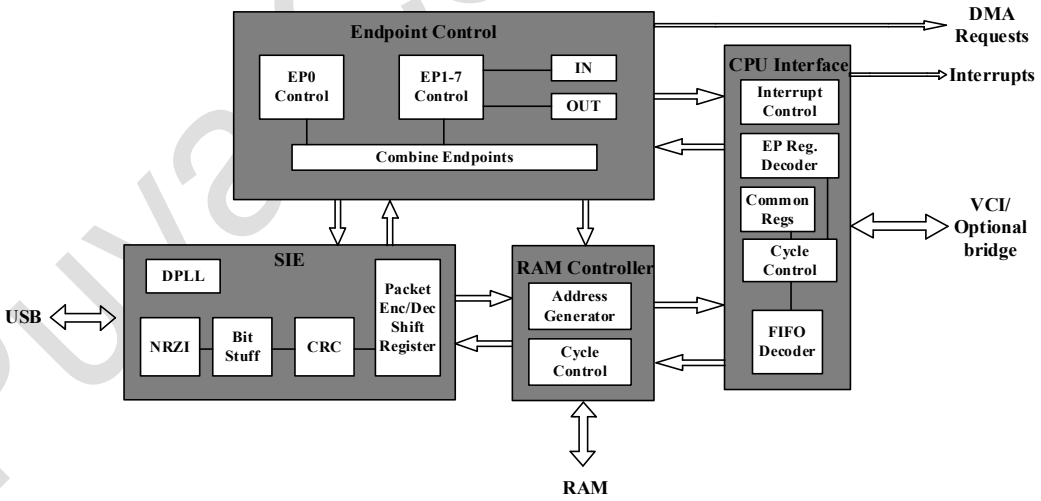


Figure 23-1 USB Module Block Diagram

23.3.2. Function Description

The USB module provides a USB specification-compliant communication link between the PC host and the functions implemented by the microcontroller. data transfer between the PC host and the microcontroller is accomplished by sharing a dedicated data buffer that can be accessed directly by the USB peripheral. The size of this dedicated data buffer is fixed and each endpoint can use a maximum of 64 bytes of buffer, with endpoint 0 using a 64-byte buffer, the rest of the IN endpoints

using a 64-byte buffer, and the OUT endpoints using a 64-byte buffer, with the same IN and OUT buffer entry address for the same endpoint. The USB module communicates with the PC host and implements the detection of packets, the processing of data transmission/reception, and the processing of handshake packets according to the USB specification.

When a valid token packet is recognized by the USB module, the associated data transfer takes place (if data transfer is required and the endpoint is configured). the USB module implements the data exchange between the port and the dedicated buffer through an internal register. After all data transfers have been completed, appropriate handshake packets are sent or received, if required, depending on the direction of the transfer.

At the end of the data transfer, the USB module triggers an interrupt associated with the endpoint. By reading the status register or by using the different interrupt handlers, the microcontroller can determine:

- Which endpoint needs to be served
- Which type of transmission was in progress when errors such as bit fill, format, CRC, protocol, missing ACK, buffer overflow/unfilled buffer, etc. were generated.

The USB module can be put in low-power mode (SUSPEND mode) by writing the control registers at any time when the USB module is not needed. In this mode, the USB internal data clock is stopped. The USB module can be woken up in low-power mode by detecting the data transfer on the USB line.

The USB module uses a fixed clock, which is defined by the USB standard as 48 MHz. the APB1 bus clock can be greater or less than this frequency.

23.3.2.1. USB Function Module Description

The USB module implements all the features of the standard USB interface, which consists of the following components:

Serial Interface Engine (SIE):The SIE handles NRZI encoding/decoding, bit padding/de-padding, and CRC generation/checking. It generates a 12MHz USB clock from the input 48MHz clock, and the data stream is synchronized under this clock when data is received from the USB host. It generates headers for the transmitted packets and decodes them when they are received.

ENDPOINT CONTROLLERS: Two state controllers are used, one for control transfers on endpoint 0 and the other for bulk, interrupt or synchronous transfer transactions on endpoints 1-7.

CPU INTERFACE: The CPU interface allows access to the control/status registers of each endpoint and the FIFO used as a data buffer. it generates an interrupt to the CPU when a packet is successfully transmitted or received, or when the host enters SUSPEND mode or wakes up from SUSPEND mode.

RAM CONTROLLER: The RAM controller provides an interface to a single block of synchronous single-port RAM, which is used to cache packets between the CPU and the USB. It takes FIFO pointers from the endpoint controller and converts them to address pointers within the RAM block and generates RAM access control signals.

23.3.3. Function Use

23.3.3.1. System reset and power-on reset

When a system reset or power-on reset occurs, the first thing the application needs to do is to provide the clock signal required by the USB module, and then clear the reset signal so that the program can access the registers of the USB module. The initialization process after a reset is described as follows:

First, the clock of the register unit is activated by the application;

Second, the relevant control bits of the device clock management logic unit are then configured;

Finally, the reset signal is cleared.

When the system is reset, the application should initialize all the registers needed to enable the USB module to generate normal interrupts and complete data transfers. All registers not related to endpoints need to be initialized according to the needs of the application (e.g., selection of interrupt enable, etc.). Next, it enters the USB reset state.

23.3.3.2. USB reset (RESET interrupt)

When a USB reset occurs, the USB module enters the state described below:

- position 0 of ADD[6:0] of the USB_CR register;
- INDEX location 0 of the USB_FRAME register;
- Clear all data in the FIFO;
- clearing all control/status registers;
- Enabling all interrupts except suspend;
- Generate a USB reset interrupt.

When the software receives a reset interrupt, it should close all open pipes and wait for the bus enumeration to start.

23.3.3.3. USB suspend/Wake up mode

When the USB device is idle for all 3ms and the enable suspend bit in the USB_CR register has been set to 1, the USB module will enter suspend mode. If the suspend interrupt is enabled (EN_Suspend bit in the USB_INTRE register), then a suspend interrupt will be generated at this point.

When the USB enters suspend mode, the internal 12MHz data clock will be stopped to reduce power consumption. The output signal named USB_SUSPEND will be set to 1, which can be used to shut down the USB driver. At the same time, the incoming 48MHz system clock needs to be maintained so that the USB module can detect signals on the bus. Of course, the input 48M system clock can be stopped to further reduce power consumption, but if the 48M system clock is stopped, the USB module will not be able to detect signals on the bus and the user will have to restart the system clock.

When the USB module is in suspend mode, any activity on the bus (non-idle signal) can wake the device up, thus exiting suspend mode and generating a resume interrupt. Of course it is also possible to force the device out of suspend mode by setting the Resume bit in the USB control register by the CPU, in which case the CPU should clear the bit after 10 ms (15 ms max). At this point the USB module is completely out of suspend mode and no resume interrupts are generated.

23.3.3.4. IN group (For data sending)

The FIFO sizes for IN endpoints 1-7 are all fixed at 64 bytes, but the maximum packet size for IN transmission is configurable and is determined by the InMaxP register written to each endpoint.

Since each packet to be sent is loaded into the IN FIFO, the InPktRdy bit in the USB_INEPCSR register needs to be set. If the AutoSet bit in the USB_INEPCSR register is set, the InPktRdy bit is automatically set when the maximum packet is loaded in the FIFO. For packets smaller than the maximum value, InPktRdy is required to be set manually (by the CPU).

When the InPktRdy bit is set, the FIFONotEmpty bit in the USB_INEPCSR register is also set, and then the packet is ready to be sent.

When the packet is successfully sent, the InPktRdy bit and the FIFONotEmpty bit are cleared, and if the corresponding interrupt is enabled, then an interrupt is generated for the corresponding IN endpoint. The next packet is then able to be loaded into the FIFO.

DOUBLE PACKET BUFFERING

If the FIFO size of the IN endpoint is at least twice the maximum packet size of this endpoint (depending on the size of InMaxP), then both packets can be cached in the FIFO.

When the first packet is loaded into the FIFO, InPktRdy is set and then immediately cleared, generating the corresponding IN endpoint interrupt. The second packet can then be loaded into the FIFO, while InPktRdy will be set again (manually or automatically) so that both packets can be sent.

When the first packet is successfully sent, the clearing of the InPktRdy bit and the generation of the corresponding IN endpoint interrupt indicates that the other packet is now ready to be loaded into the FIFO. The status of the FIFONotEmpty bit at this point indicates how many packets may have been loaded. If FIFONotEmpty is set, then another packet will be in the FIFO and only one packet will be loaded. If the FIFONotEmpty bit is cleared, there are no packets in the FIFO and more than two packets can be loaded.

23.3.3.5. OUT grouping (for data reception)

The FIFO sizes for OUT endpoints 1 through 7 are fixed at 64 bytes, but the maximum packet size for OUT transfers is programmable and is determined by the OutMaxP register written to each endpoint.

When a packet is received and placed in the OUT FIFO, the OutPktRdy bit and the FIFOFull bit in USB_OUTEPCSR are set and the appropriate OUT endpoint interrupt is generated to indicate that a packet is now ready to be read from the FIFO. The OutPktRdy bit needs to be cleared after the packet has been read in order to receive more packets. If the AutoClear bit in USB_OUTEPCSR is set and the largest packet is read away from the FIFO, the OutPktRdy bit is automatically cleared. the FIFOFull bit is also cleared. For packets smaller than the maximum packet size, OutPktRdy always has to be cleared manually (i.e. by the CPU).

Double packet buffering

The OUT FIFO can cache two packets if the size of the OUT endpoint FIFO is at least twice the maximum packet size for that endpoint (set in the OutMaxP register).

When the first packet to be received is loaded into the OUT FIFO, the OutPktRdy bit in USB_OUTEPxCSR is set and the corresponding OUT endpoint interrupt is generated to indicate that the packet is now ready to be read from the FIFO.

Note: The FIFOFull bit in USB_OUTEPCSR is not set at this point in time; it is only set when the second packet is received and loaded into the OUT FIFO.

After reading away the first packet, the OutPktRdy bit needs to be cleared in order to receive more packets. If the AutoClear bit in USB_OUTEPCSR is set and the maximum size packet is read away from the FIFO, the OutPktRdy bit will be automatically cleared. For packets smaller than the maximum packet size, OutPktRdy always has to be cleared manually (i.e. by the CPU).

If the FIFOFull bit is set to 1 when OutPktRdy is cleared, the USB module will first clear the FIFOFull bit. Then the OutPktRdy bit is set again to indicate that there is another packet in the FIFO waiting to be read away.

23.3.3.6. Control Transmission

Endpoint 0 is the primary control endpoint for USB. Therefore, the routines required to drive endpoint 0 are more complex than those required to drive the other endpoints.

Software can receive and process all standard device requests through endpoint 0. These are described in the USB Bus Specification. The protocol for these device requests involves a different number and type of transactions per transfer. To accommodate this, the CPU needs to use a state machine approach to decode and process the commands. Standard device requests can be divided into three categories: zero data requests, write requests and read requests. This section describes the sequence of events that must be executed by the software to handle the different types of device requests.

Note:The Setup package associated with any standard device request should include an 8-byte command. Any Setup package that contains more than 8 bytes of command characters will be automatically rejected by the USB host.

Zero data requests

All information for zero-data requests is contained in the 8-byte command and no additional data needs to be transferred. Examples of zero data standard device requests are:SET_FEATURE, CLEAR_FEATURE, SET_ADDRESS, SET_CONFIGURATION, SET_INTERFACE.

As with all requests, the event sequence will begin when the software receives an Endpoint 0 interrupt. the OutPktRdy bit (in the USB_EP0CSR register) will also be set. the 8-byte command should be read from the Endpoint 0 FIFO, decoded and the appropriate action taken. For example, if the command is SET_ADDRESS, the 7-bit address value contained in the command should be written to the FAddr register.

The USB_EP0CSR register should then be written to set the ServicedOutPktRdy bit (indicating that the command has been read from the FIFO) and to set the DataEnd bit (indicating that no further data is needed for the request).

When the host moves to the status phase of the request, a second endpoint 0 interrupt is generated to indicate that the request has completed. No further action is required by the software; the second interrupt simply confirms that the request completed successfully.

If the command is an unrecognized command, or cannot be executed for some other reason, the USB_EP0CSR register should be written to set the ServicedOutPktRdy bit and set the SendStall bit when it is decoded. When the host moves to the status stage of the request, the device will send a STALL to tell the host that the request is not being executed. This will generate a second endpoint 0 interrupt and set the SentStall bit.

If the host sends more data after the DataEnd bit is set, then the device will send a STALL. this will generate an endpoint 0 interrupt and the SentStall bit will be set.

Write requests

A write request consists of one (or more) additional packets that are sent from the host after the 8-byte command. An example of a write standard device request is :SET_DESCRIPTOR.

As with all requests, the event sequence will begin when the software receives an endpoint 0 interrupt. the OutPktRdy bit (in the USB_EP0CSR register) will also be set. the 8-byte command should be read and decoded from the endpoint 0 FIFO.

As with zero data requests, the ServicedOutPktRdy bit in the USB_EP0CSR register should then be set (indicating that the command has been read from the FIFO), but the DataEnd bit should not be set in this case (indicating that more data is needed).

When the second endpoint 0 interrupt is received, the USB_EP0CSR register should be read to check the endpoint status. the OutPktRdy bit should be set to indicate that a packet has been received. The COUNT0 register should then be read to determine the size of this packet. The packet can be read from the endpoint 0 FIFO.

If the length of the data associated with the request (indicated by the wLength field in the command) is greater than the maximum packet size for endpoint 0, further packets will be sent. In this case, USB_EP0CSR should be written to set the ServicedOutPktRdy bit, but the DataEnd bit should not be set.

When all expected packets are received, the ServicedOutPktRdy bit and the DataEnd bit in the USB_EP0CSR register should be set up (indicating that no more data is needed).

When the host moves to the status phase of the request, another endpoint 0 interrupt is generated to indicate that the request has been completed. No further action is required by the software; the interrupt is simply an acknowledgement that the request has completed successfully.

If the command is an unrecognized command or cannot be executed for some other reason, the ServicedOutPktRdy bit and the SendStall bit in the USB_EP0CSR register should be set when it is decoded. When the host sends more data, the USB device will send a STALL to tell the host that the request was not executed. This will generate an endpoint 0 interrupt and the SentStall bit will be set.

If the host sends more data after setting DataEnd, then the USB device will send a STALL. an endpoint 0 interrupt will be generated and the SentStall bit will be set up.

Read requests

After the 8-byte command, the read request has one (or more) packets sent from the function (function) to the host. Examples of standard device requests are: GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, GET_STATUS, SYNCH_FRAME.

As with all requests, the event sequence will start when the software receives an endpoint 0 interrupt. the OutPktRdy bit (in the USB_EP0CSR register) will also be set. the 8-byte command should be read from the endpoint 0 FIFO and decoded. The ServicedOutPktRdy bit in the USB_EP0CSR register should then be set (indicating that the command has been read from the FIFO).

The data sent to the host should be written to the FIFO of endpoint 0. If the data to be sent is larger than the maximum packet size of endpoint 0, only the maximum packet size should be written to the FIFO. then the InPktRdy bit in register USB_EP0CSR is set (indicating that there is a packet to be sent in the FIFO). When the packet is sent to the host, another endpoint 0 interrupt will be generated and the next packet can be written to the FIFO.

When the last packet is written to the FIFO, the InPktRdy bit and the DataEnd bit in the USB_EP0CSR register should be set (indicating that there is no more data after this packet).

When the host moves to the status phase of the request, another endpoint 0 interrupt will be generated to indicate that the request is complete. No further action is required by the software; the interrupt is simply an acknowledgement that the request has completed successfully.

If the command is an unrecognized command or cannot be executed for some other reason, the ServicedOutPktRdy bit and the SendStall bit in the USB_EP0CSR register should be set when it is decoded. When the host requests data, the USB device will send a STALL to tell the host that the request was not executed. An endpoint 0 interrupt will then be generated and the SendStall bit will be set.

If the host requests more data after DataEnd is set, then the USB device will send a STALL. an Endpoint 0 interrupt will be generated and the SentStall bit will be set.

Error handling

A control transfer on a USB module may be aborted due to a protocol error and the host will terminate the transfer early, or if the software function controller (function con-troller software) wishes to terminate the transfer (e.g., because it cannot handle the command). USB automatically detects a protocol error and sends STALL to the host in the following cases.

1) The host sends more data than specified in the command during the OUT data phase of the write request. This condition is detected when the host sends the OUT token after the DataEnd bit is set.
2) The host requests more data than specified in the command during the IN data phase of the read request. This condition will be detected when the host sends an IN token after setting the DataEnd bit in the USB_EP0CSR register.
3) The host sends an OUT packet with more than MaxP data bytes.
4) The host sends a non-zero length DATA1 packet during the STATUS phase of the read request.

When a USB device sends a STALL packet, it places the SentStall bit and generates an interrupt. When software receives an endpoint 0 interrupt with SentStall bit set, it should abort the current transmission, clear the SentStall bit, and return to the idle state.

If the host enters the state phase before all data transfer of the transfer request is complete, or if it sends a new SETUP packet before completing the current transfer, thus ending the transfer early, then the SetupEnd bit is set and an Endpoint 0 interrupt is generated. When software receives an endpoint 0 interrupt with the SetupEnd bit set, it should abort the current transmission, set the ServicedSetupEnd bit, and return to the idle state. If the OutPktRdy bit is set, this indicates that the host has sent another SEPUP packet and then the software should process the command.

If the software cannot process the command or has other internal errors and wishes to terminate the current transfer, it should set the SendStall bit. The USB device will then send a STALL packet to the host with the SendStall bit in place and generate an endpoint 0 interrupt.

Synchronous transmission

The USB protocol defines a full-speed transmission method that requires maintaining a fixed and precise data transfer rate: synchronous transmission. Synchronous transfers are typically used to transfer audio streams, compressed video streams, and other data with strict data transfer rate requirements. If an endpoint is defined as a "synchronous endpoint" during enumeration, the USB host allocates a fixed amount of bandwidth to each frame and ensures that each frame transmits exactly one IN packet or OUT packet (the type of packet is determined by the direction of the endpoint transmission). To meet the bandwidth requirements, there are no error retransmissions in synchronous transfers; this also means that synchronous transfers send or receive packets with no handshake protocol, i.e., no ACK packets are sent. Similarly, synchronous transmission only transmits packets with PID DATA0, and no data flipping mechanism is used.

23.3.3.7. Synchronous read transfer (ISOCHRONOUS IN ENDPOINT)

Synchronous read transfer is used to transfer periodic data from the USB module to the host. Three optional features can be used to synchronize IN endpoints.

- Double packet caching function

  Double packet buffering is automatically enabled when the value written to the InMaxP register is less than or equal to half the size of the FIFO allocated to the endpoint. When enabled, up to two packets can be stored in the FIFO waiting to be transferred to the host.

- DMA function

  If the endpoint is DMA enabled, a DMA request is generated whenever the endpoint is able to receive another packet in its FIFO. This feature can be used to allow an external DMA controller to load packets into the FIFO without processor intervention.

- AutoSet function

  When the AutoSet function is enabled, the InPktRdy bit will be automatically set when an InMaxP byte packet is loaded into the FIFO.

Before a synchronous IN endpoint can be used, the InMaxP register must be written with the maximum packet size (in bytes) for that endpoint. This value should be the same as the wMaxPacketSize field of the endpoint's standard endpoint descriptor. In addition, the associated interrupt enable bit in the USB_INTRE register should be set to 1 and some of the bits in the USB_INEPCSR register should be set as shown below.

Table 23-1 Partial bit setting in USB_INEPCSR register

AutoSet	ISO	Mode	DMAEnab	FrcDataTog
0/1	1	1	0/1	0

A synchronous endpoint does not support data retransmission, so the data sent to the host must be loaded into the FIFO before the IN token is received if data errors are to be avoided. The host will send one IN token per frame, but the time within the frame is variable. If an IN token is received at the end of one frame and then at the beginning of the next frame, there will be little time to reload the FIFO. for this reason, a double cache is usually required in the synchronous IN endpoint.

The AutoSet feature can be used to synchronize IN endpoints, but unless the data from the source arrives at an absolutely consistent rate and is synchronized with the host's frame clock, the size of the packet sent to the host will have to increase or decrease frame by frame to match the source data rate. This means that the actual packet size is not always the InMaxP size, which makes the AutoSet feature useless.

An interrupt is generated when a packet is sent to the host, and software can use this interrupt to load the next packet into the FIFO and set the InPktRdy bit in the USB_INEPCSR register. Since interrupts can occur at almost any time within a frame, depending on when the host schedules the routine, this can lead to irregular timing of FIFO load requests. If the endpoint's data source is from some external hardware, it may be more convenient to wait for the end of each frame before loading the FIFO, as this will minimize the need for additional buffering. The above operation can be achieved by using the SOF interrupt or an external SOF_PULSE signal from the USB device to trigger the next packet load. When a SOF packet is received, a SOF_PULSE signal is generated per frame (the USB also keeps an external frame counter, so it can still generate SOF_PULSE when a SOF packet is lost). This SOF_PULSE interrupt can still be used to set the InPktRdy bit in USB_INEPCSR and to check for data overflow/missing.

The synchronous IN pipeline with double caching enabled can be a source of problems. Dual caching requires that packets be transmitted only after the frames are loaded. If the feature loads the first packet at least one frame before the host builds the pipeline (and then starts sending IN tokens), there is no problem. However, if the host already starts sending IN tokens when the first packet is loaded, the packet may be transmitted in the same frame as it is loaded, depending on whether it is loaded before or after the IN token is received. This potential problem can be avoided by setting the ISO Update bit in the USB_CR register. When this bit is set to 1, any packets loaded into the synchronous IN endpoint FIFO will not be sent until the next SOF packet is received, thus ensuring that packets are not sent prematurely.

If the endpoint has no data in the FIFO when it receives the IN token, it will send an empty packet to the host and set the UnderRun bit in the USB_INEPxCSR register. This indicates that the software is not providing the host with data fast enough. It is up to the application to decide how to handle this error condition.

If the software finds that the InPktRdy bit in the USB_INEPxCSR register is set when it wants to load the next packet in each frame, this indicates that the packet has not been sent yet (perhaps because an IN token packet from the host is corrupted). It depends on how the application handles this situation: it can choose to flush the unsent packet by setting the FlushFIFO bit in the USB_INEPxCSR register, or it can choose to skip the current packet.

Synchronous write transfer (ISOCHRONOUS OUT ENDPOINT)

The synchronous OUT endpoint is used to transmit periodic data from the functional controller to the host. Three optional features can be used to synchronize the OUT endpoints.

- Double packet caching function

  Double packet buffering is automatically enabled when the value written to the OutMaxP register is less than or equal to half the size of the FIFO allocated to the endpoint. When enabled, up to two packets can be stored in the FIFO.

- DMA function

  If the endpoint is DMA enabled, a DMA request is generated whenever a packet is in the endpoint's FIFO. This feature can be used to allow an external DMA controller to read packets from the FIFO without processor intervention.

- AutoClear function

  When the AutoClear function is enabled, the OutPktRdy bit will be automatically cleared when the packet of OutMaxP byte is read from the FIFO.

Before using the synchronous OUT endpoint, the OutMaxP register must be written with the maximum packet size (in bytes) of the endpoint. This value should be the same as the wMaxPacketSize field of the endpoint's standard endpoint descriptor. In addition, the relevant interrupt enable bit in the USB_INTRE register should be set to 1 and some of the bits in the USB_OUTEPxCSR register should be set as shown below.

Table 23-2 Partial bit setting in USB_OUTEPxCSR register

AutoClear	ISO	DMAEnab
0/1	1	0/1

A synchronous endpoint does not support data retransmission, so there must be room in the FIFO to receive packets if data overflow is to be avoided. The host will send one packet per frame, but the time within the frame can vary. If a packet is received at the end of one frame and another arrives at the beginning of the next frame, there is little time to read the data in the FIFO. Therefore, for synchronous OUT endpoints, a dual packet caching feature is usually required.

The AutoClear feature can be used with synchronous OUT endpoints. However, unless the data receiver receives data at an absolutely consistent rate and synchronizes with the host's frame clock, the packet size sent by the host will have to increase or decrease frame by frame to match the desired data rate. This means that the actual packet size is not always the OutMaxP size, which renders the AutoClear feature useless.

When a packet is received from the host, an interrupt is generated that software can use to read the packet away from the FIFO and clear the OutPktRdy bit in the USB_OUTEPCSR register. Since interrupts can occur at almost any time within a frame, depending on when the host has scheduled the transaction, the timing of FIFO read data requests can be irregular. If the endpoint's data receiver is to transmit to some external hardware, it is best to wait for the end of each frame before reading the FIFO, thus reducing the need for additional buffering. This can be done by using SOF interrupts or an external SOF_PULSE signal from the USB device to trigger the packet read. The SOF_PULSE signal is generated once per frame when a SOF packet is received (the USB device also keeps an external frame counter so it can still generate SOF_PULSE when a SOF packet is

lost). This interrupt can still be used to clear the OutPktRdy bit in USB_OUTEPxCSR and check for data overflow/missing.

If there is no space in the FIFO to store the packets received from the host, the overflow bit in the USB_OUTEPxCSR register will be set. This indicates that the software is not reading the data fast enough. It is up to the application to decide how to handle this error condition.

If the USB device finds a received packet with a CRC error, it still stores the packet in the FIFO and sets the OutPktRdy bit and the DataError bit. It is up to the application to decide how to handle this error condition.

23.3.3.8. Bulk Transfer

Batch read transfer (BULK IN ENDPOINT)

The Bulk IN endpoint is used to transfer irregular data from the function controller to the host. There are three optional features available for the Bulk IN endpoint.

- Double packet caching function

  If the value written to the InMaxP register is less than or equal to half the size of the FIFO allocated to the endpoint, the dual packet caching feature is automatically enabled. When enabled, up to two packets can be stored in the FIFO pending transmission to the host.

- DMA function

  If the endpoint is DMA enabled, a DMA request is generated whenever the endpoint is able to accept another packet in its FIFO. This feature can be used to allow an external DMA controller to load packets into the FIFO without processor intervention.

- AutoSet function

  When the AutoSet feature is enabled, the InPktRdy bit will be automatically set when an InMaxP byte packet is loaded into the FIFO. This is particularly useful when loading the FIFO using DMA, as it avoids the need for processor intervention when loading individual packets during bulk transfers.

The maximum packet size (in bytes) of the endpoint must be used to write to the InMaxP register before the Bulk IN endpoint can be used. This value should be the same as the wMaxPacketSize field of the endpoint's standard endpoint descriptor. In addition, the associated interrupt enable bit in the USB_INTRE register should be set to 1 and the USB_INEPCSR register should be set as shown below.

Table 23-3 Partial bit setting in USB_INEPCSR register

AutoSet	ISO	Mode	DMAEnab	FrcDataTog
0/1	0	1	0/1	0

When first configuring a Bulk IN endpoint, the USB_INEPCSR register should be written to set the ClrDataTog bit after executing the SET_CONFIGURATION or SET_INTERFACE command on endpoint 0. This will ensure that the data switch is initiated in the correct state. Also, if there are any packets in the FIFO (indicated by the set FIFONotEmpty bit), they should be flushed by setting the FlushFIFO bit.

Note: If the dual packet caching feature is enabled, it may be necessary to set this bit twice in succession.

When data is to be transferred through the Bulk IN pipeline, a packet needs to be loaded into the FIFO and the InPktRdy bit in the USB_INEPxCSR register set. When the packet is sent, the InPktRdy bit is cleared and an interrupt is generated so that the next packet can be loaded into the FIFO. If dual packet caching is enabled, then as soon as the first packet is loaded and the InPktRdy bit is set, the InPktRdy bit will be cleared and an interrupt will be generated so that the second packet can be loaded into the FIFO. The software should operate in the same way, loading a packet when it receives an interrupt, whether or not the dual packet cache is enabled.

The size of the packet cannot exceed the size specified in the InMaxP register. When a block of data larger than InMaxP is to be transmitted, it must be sent as multiple packets. The size of these packets should be the value set by InMaxP, except for the last packet. The host can determine that all data transmitted has been sent by knowing the total amount of data expected. Or, when it receives a packet smaller than the InMaxP size, it can infer that all the data has been sent. In the latter case, if the total size of the data block is a multiple of InMaxP, then the function (function) needs to send an empty packet after all the data has been sent. This is done by setting InPktRdy when the next interrupt is received, without loading any data into the FIFO.

If large amounts of data are being transferred, then by using DMA you can avoid calling the interrupt service program to load each packet.

If the software wants to close the Bulk IN pipeline, it should set the SendStall bit. When the USB device receives the next IN token, it will send a STALL to the host, setting the SendStall bit and generating an interrupt.

When the software receives an interrupt with the SentStall bit set, it should clear this SentStall bit. Of course, it should also keep the SendStall bit set until it is ready to re-enable the Bulk IN pipeline.

Note:If the host is unable to receive the STALL packet for some reason, it will send another IN token, so it is recommended to leave the SendStall bit set until the software is ready to re-enable the Bulk IN pipeline. When the pipeline is re-enabled, the data switching sequence should be restarted by setting the ClrDataTog bit in the USB_INEPCSR register.

Bulk write transfer (BULK OUT ENDPOINT)

The Bulk OUT endpoint is used to transfer unscheduled data from the host to the USB module. There are three optional features available for the Bulk OUT endpoint.

- Double packet caching function

  If the value written to the OutMaxP register is less than or equal to half the size of the FIFO allocated to the endpoint, then double packet buffering will be automatically enabled. When enabled, up to two packets can be stored in the FIFO.

- DMA function

  If the endpoint is DMA enabled, a DMA request is generated whenever a packet is in the endpoint's FIFO. This feature can be used to allow external DMA controllers to read packets from the FIFO without processor intervention.

- AutoSet function

  When the AutoClear feature is enabled, the OutPktRdy bit is automatically cleared when packets of OutMaxP bytes are offloaded from the FIFO. This is especially useful when reading

data from the FIFO using DMA. This is because it avoids the processor intervention required when reading individual packets during large Bulk transfers.

Before using the Bulk OUT endpoint, the maximum packet size (in bytes) of the endpoint must be used to write to the OutMaxP register. This value should be the same as the wMaxPacketSize field of the endpoint's standard endpoint descriptor. In addition, the relevant interrupt enable bit in the USB_INTRE register should be set to 1 (if this endpoint requires an interrupt) and the relevant bit in the USB_OUTEPxCSR register should be set as shown below.

Table 23-4 Partial bit setting in USB_OUTEPxCSR register

AutoClear	ISO	DMAEnab
0/1	0	0/1

When first configuring a Bulk OUT endpoint, the USB_OUTEPCSR should be written to set the ClrDataTog bit after the SET_CONFIGURATION or SET_INTERFACE command on endpoint 0. This will ensure that the data switch is initiated in the correct state. Similarly, if there are any packets in the FIFO (OutPktRdy bit is set), they should be flushed by setting the FlushFIFO bit.

Note:If double packet caching is enabled, it may be necessary to set this bit twice in succession.

When a packet is received by a Bulk OUT endpoint, the OutPktRdy bit is set and an interrupt is generated. The software should read the OutCount register of the endpoint to determine the packet size. The packet should be read from the FIFO first, then the OutPktRdy bit should be cleared.

The packet size should not exceed the size specified in the OutMaxP register (as this should be the value set in the wMaxPacketSize field of the endpoint descriptor sent to the host). When a block of data larger than OutMaxP is to be sent to a function, it will be sent as multiple packets. The size of all packets is the size set by OutMaxP, except for the last packet, which will contain the remainder. Software can use application-specific methods to determine the total size of the block to determine when the last packet is received. Alternatively, when it receives a packet smaller than the size of OutMaxP, it may infer that the entire data has been received (if the total size of the data block is a multiple of OutMaxP, an empty packet will be sent at the end of the data, indicating that the transmission is complete).

If a large amount of data is being transferred, by using DMA you can avoid calling the interrupt service program to read the data for each packet.

If the software wants to close the Bulk OUT pipeline, it should set the SendStall bit. When the USB device receives the next packet, it will send a STALL to the host, setting the SentStall bit and generating an interrupt.

When the software receives an interrupt with the SentStall bit set, it should clear this SentStall bit. It should also leave the SendStall bit set until it is ready to re-enable the Bulk OUT pipeline.

Note:If the host fails to receive a STALL packet for some reason, it will send another packet, so it is recommended that the SendStall bit be retained until the software is ready to re-enable the Bulk OUT pipeline. When re-enabling the Bulk OUT pipeline, the data switching sequence should be restarted by setting the ClrDataTog bit in the USB_OUTEPxCSR register.

23.3.3.9. Interrupted transmission

Interrupted Read Transfer (INTERRUPT IN ENDPOINT)

The Interrupt IN endpoint is used to transmit periodic data from the function controller to the host.

The Interrupt IN endpoint uses the same protocol as the Bulk IN endpoint and can be used in the same way. Although DMA can be used, it provides few benefits because the Interrupt IN endpoint typically expects to transfer all data in a single packet.

Interrupt IN endpoints also support a feature that Bulk IN endpoints do not support, namely they support continuous switching of data switching bits. This feature is enabled by setting the FrcDataTog bit in the USB_INEPCSR register. When this bit is set to 1, the USB device will assume that the packet has been successfully sent and toggle the data bits for the endpoint, regardless of whether an ACK is received from the host.

Interrupt write transfer (INTERRUPT OUT ENDPOINT)

The Interrupt OUT endpoint is used to transfer periodic data from the host to the function controller.

The Interrupt OUT endpoint uses almost the same protocol as the Bulk OUT endpoint and can be used in the same way. Although DMA can be used with an interrupt OUT endpoint, it typically provides little benefit because the interrupt endpoint typically expects to transfer all of the data in a packet.

23.3.4. USB Register Description

Note: When writing to USB registers, the operation can only be done byte or word, and when reading to USB registers, the operation can only be done byte.

23.3.4.1. USB control register (USB_CR)

Address offset：

0x00

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISO Update	Res			Reset	Resume	Suspend Mode	Enable Suspend	Update	ADD						
RW				R	RW	R	RW	R	RW						

Bit	Name	R/W	Reset Value	Function
31:16	Reserved			
15	ISO Update	RW	0	After this position 1, the USB controller needs to wait for a SOF after InPktRdy is set to 1 before sending a packet; if the IN Token is received before the SOF is received, the USB controller will send a zero packet. (This register is only used during ISO transmission)
14:12	Reserved			
11	Reset	R	0	When the USB bus has a reset signal, this bit is set to 1
10	Resume	RW	0	When the USB device is in Suspend mode, the software sets 1 to

				generate the Resume wake-up signal; the software clears the bit after 10mS. (Maximum 15mS)
9	Suspend Mode	R	0	When entering Suspend mode, the bit is set to 1 by the USB device hardware; it is cleared when the interrupt register is read by software, or when the Resume register is written by software
8	Enable Suspend	RW	0	Suspend function is enabled
7	Update	R	0	Set to 1 when Func Addr is written; cleared when the address takes effect (at the end of the transfer)
6:0	ADD	RW	0	Function Address

23.3.4.2. USB Interrupt Status Register (USB_INTR)

Address offset：0x04

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								EP7IN	EP6IN	EP5IN	EP4IN	EP3IN	EP2IN	EP1IN	EP0
								R	R	R	R	R	R	R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP7OUT	EP6OUT	EP5OUT	EP4OUT	EP3OUT	EP2OUT	EP1OUT	Reserved					SOF	Reset	Resume	Suspend
R	R	R	R	R	R	R						R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:24	Reserved			
23	EP7IN	R	0	IN Endpoint 7 Interrupt
22	EP6IN	R	0	IN Endpoint 6 Interrupt
21	EP5IN	R	0	IN Endpoint 5 Interrupt
20	EP4IN	R	0	IN Endpoint 4 Interrupt
19	EP3IN	R	0	IN Endpoint 3 Interrupt
18	EP2IN	R	0	IN Endpoint 2 Interrupt
17	EP1IN	R	0	IN Endpoint 1 Interrupt
16	EP0	R	0	IN Endpoint 0 Interrupt
15	EP7OUT	R	0	OUT Endpoint 7 Interrupt
14	EP6OUT	R	0	OUT Endpoint 6 Interrupt
13	EP5OUT	R	0	OUT Endpoint 5 Interrupt
12	EP4OUT	R	0	OUT Endpoint 4 Interrupt
11	EP3OUT	R	0	OUT Endpoint 3 Interrupt
10	EP2OUT	R	0	OUT Endpoint 2 Interrupt
9	EP1OUT	R	0	OUT Endpoint 1 Interrupt
8:4	Reserved			
3	SOF	R	0	Set 1 at the beginning of each frame
2	Reset	R	0	Set 1 when a reset signal is detected on the USB bus
1	Resume	R	0	When the USB device is in Suspend mode, set 1 when the Resume signal is detected on the USB bus
0	Suspend	R	0	Set 1 when the Suspend signal is detected on the USB bus

23.3.4.3. USB interrupt enable register (USB_INTRE)

Address offset：0x08

Reset value：0x00FFFE06

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								EP7INE	EP6INE	EP5INE	EP4INE	EP3INE	EP2INE	EP1INE	EP0E
								RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EP7OUTE	EP6OUTE	EP5OUTE	EP4OUTE	EP3OUTE	EP2OUTE	EP1OUTE	Reserved					EN_SOF	EN_Reset	EN_Resume	EN_Suspend
RW	RW	RW	RW	RW	RW	RW						RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:24	Reserved			
23	EP7INE	RW	1	IN Endpoint 7 Interrupt Enable
22	EP6INE	RW	1	IN Endpoint 6 Interrupt Enable
21	EP5INE	RW	1	IN Endpoint 5 Interrupt Enable
20	EP4INE	RW	1	IN Endpoint 4 Interrupt Enable
19	EP3INE	RW	1	IN Endpoint 3 Interrupt Enable
18	EP2INE	RW	1	IN Endpoint 2 Interrupt Enable
17	EP1INE	RW	1	IN Endpoint 1 Interrupt Enable
16	EP0E	RW	1	IN Endpoint 0 Interrupt Enable
15	EP7OUTE	RW	1	OUT Endpoint 7 Interrupt Enable
14	EP6OUTE	RW	1	OUT Endpoint 6 Interrupt Enable
13	EP5OUTE	RW	1	OUT Endpoint 5 Interrupt Enable
12	EP4OUTE	RW	1	OUT Endpoint 4 Interrupt Enable
11	EP3OUTE	RW	1	OUT Endpoint 3 Interrupt Enable
10	EP2OUTE	RW	1	OUT Endpoint 2 Interrupt Enable
9	EP1OUTE	RW	1	OUT Endpoint 1 Interrupt Enable
8:4	Reserved			
3	EN SOF	RW	0	SOF Interrupt Enable
2	EN Reset	RW	1	Reset Interrupt Enable
1	EN Resume	RW	1	Resume Interrupt Enable
0	EN Suspend	RW	0	Suspend Interrupt Enable

23.3.4.4. USB_FRAME

Address offset：0x0C

Reset value：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												INDEX			
												RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					FramNUM										
					R										

Bit	Name	R/W	Reset Value	Function
31:20	Reserved			
19:16	INDEX	RW	0	Endpoint selection register
15:11	Reserved			
10:0	FramNUM	RW	0	Last received Fram Number

23.3.4.5. USB Endpoint 0 Status Control Register (USB_EP0CSR)

Address offset：0x10

Reset value：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	COUNT0							ServicedSetupEnd	ServicedOutPktRdy	SendStall	SetupEnd	DataEnd	SentStall	InPktRdy	OutPktRdy
	R							W1	W1	W1	R	W1	RW0	RW1	R

Bit	Name	R/W	Reset Value	Function
31:15	Reserved			
14:8	COUNT0	R	0	The length of data received by EP0, read when OutPktRdy is set to 1 is valid
7	ServicedSetupEnd	W1	0	Software write 1 to clear SetupEnd, the bit is automatically cleared
6	ServicedOutPktRdy	W1	0	Software write 1 clears OutPktRdy, the bit is automatically cleared
5	SendStall	W1	0	Software write 1 terminates the current transmission; the STALL handshake is sent, after which the bit is automatically cleared
4	SetupEnd	R	0	This bit, at the end of the control transfer, DataEnd set 1 precedes 1, will generate an interrupt and clear the FIFO
3	DataEnd	W1	0	software writes 1 and the bit is automatically cleared in the following cases; 1, after the last packet is sent and InPktRdy is set. 2, after the most packet has been read and OutPktRdy has been cleared by software; 3, after sending a zero packet, setting InPktRdy;
2	SentStall	RW0	0	Set to 1 after sending STALL handshake, cleared by software;
1	InPktRdy	RW1	0	Software Write 1 When a packet is written to the FIFO. The packet is cleared after the packet transfer is completed and an interrupt is generated after clearing
0	OutPktRdy	R	0	This bit is set to 1 when a packet is received and generates an interrupt

23.3.4.6. USB_INEPxCSR

Address offset：0x14

Reset value：0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								INMAXP							
								RW							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ClrDataTog	SentStall	SendStall	FlushFIFO	UnderRun	FIFONotEmpty	InPktRdy	AUTOSet	ISO	Mode	DMAEnable	FrcDataTog	Reserved		
	W1	RW0	RW	W1	R	RW0	RW	RW	RW	RW	RW	RW			

Bit	Name	R/W	Reset Value	Function
31:24	Reserved			
23:16	INMAXP	RW	0	The largest packet at the IN Endpoint; the maximum packet size is the value of the 8 bits * 8, one for each IN Endpoint, except EP0
15	Reserved			
14	ClrDataTog	W1	0	Software write 1, reset IN EP data toggle
13	SentStall	RW0	0	Set 1 after sending STALL handshake packet; at this time the FIFO should be cleared and InPktRdy should also be cleared, this bit is cleared by software
12	SendStall	RW	0	Software write 1, which sends a STALL handshake upon receipt of the IN Token; When the STALL environment disappears, the soft clear is done and the bit is invalid for ISO
11	Flush FIFO	W1	0	Software write 1 clear IN FIFO; can only clear the next packet to be transmitted, if there are two packets of data in the FIFO, you need to write twice, the bit is automatically cleared
10	UnderRun	R	0	In ISO mode, when a zero packet is sent after receiving an IN Token and InPktRdy is not set to 1, this position is 1, and this bit is cleared by software
9	FIFONot Empty	RW0	0	IN FIFO non-empty flag
8	InPktRdy	RW	0	When a packet is written to the FIFO, the software writes 1; the packet is cleared after the transmission is completed, and an interrupt is generated after the clearing
7	AutoSet	RW	0	After setting 1, when the data written in IN FIFO reaches the maximum packet (InMaxP), InPktRdy is automatically set to 1
6	ISO	RW	0	Set to 1 to enable ISO transmission, clear to Bulk or Interrup transmission
5	Mode	RW	0	1: IN Endpoint; 0: OUT Endpoint.
4	DMAEnable	RW	0	In Endpoint DMA Request Enable
3	FrcDataTog	RW	0	Set to 1 to force the tog signal to flip and clear the packets in the FIFO
2:0	Reserved			

23.3.4.7. USB_OUTEPxCSR

Address offset：0x18

Reset value：0x00000000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ClrData Tog	Sent Stall	Send Stall	Flus h FIFO	DataE rror	Over Run	FIFOFu ll	Out Pkt Rdy	AUT OCl ear	IS O	DMAE nable	DMA Mod e	Reserved			
W1	RW0	RW	W1	R	RW0	R	RW 0	RW	R W	RW	RW				

Bit	Name	R/W	Reset Value	Function
31:24	Reserved			
23:16	OUTMAXP	RW	0	The largest packet of the OUT Endpoint, the maximum packet size is the value of the 8 bits * 8, one for each OUT Endpoint, except EP0
15	ClrDataTog	W1	0	Software write 1 reset EP data toggle to 0
14	SentStall	RW0	0	Set to 1 after sending STALL handshake; this bit is cleared by software
13	SendStall	RW	0	software write 1 to send STALL handshake and software clear to end STALL; Not valid in ISO mode
12	Flush FIFO	W1	0	Clear OUT FIFO, write once to clear one packet of data
11	DataError	R	0	After OutPktRdy is set to 1, the packet has CRC error or bit-stuff error; automatically cleared after OutPktRdy is cleared to zero; valid only in ISO mode
10	OverRun	RW0	0	OUT packets can no longer be written to the OUT FIFO and the software is cleared; Valid only in ISO mode
9	FIFO Full	R	0	OUT FIFO full flag
8	OutPktRdy	RW	0	Set 1 after receiving the packet; after the data is read out from the FIFO, the software clears it and an interrupt is generated
7	AutoClear	RW	0	After setting 1, OutPktRdy is automatically cleared when the data read from OUT FIFO reaches the value of OutMaxP
6	ISO	RW	0	1: ISO mode; 0: Bulk or Interrupt mode;
5	DMAEnalbe	RW	0	DMA enable signal
4	DMAMode	RW	0	0: DMA requests are generated for all received packets and interrupts are generated; 1: data received from OutMaxP generates DMA requests without interrupts; other packet sizes, generate interrupts but not DMA requests;
3:0	Reserved			

23.3.4.8. USB_OUTCOUNT

Address offset：0x1C

Reset value：0x00000000

31	30	29	28	27	26	25	24	23	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					OUTCOUNT										
					R										

Bit	Name	R/W	Reset Value	Function
31:11	Reserved			
10:0	OUTCOUNT	R	0	Received data length, read valid when OutPktRdy is set to 1

23.3.4.9. USB_FIFODATA

Address offset：0x20 ~ 0x3F

Reset value：0x00000000

Note: FIFO can only be operated by word or byte

FIFODATA(0~7, each EP occupies 4 Byte addresses)
RW
0

23.3.4.10. USB Register Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USB_CR	Reserved																ISO Update	Reserved			Reset	Resume	Suspend Mode	Enable Suspend	Update	ADD						
	Read/Write																	rw				r	rw	r	rw	r	rw						
	Reset Value	0																0	0			0	0	0	0	0	0						
0x04	USB_INTR	Reserved								EP7IN	EP6IN	EP5IN	EP4IN	EP3IN	EP2IN	EP1IN	EP0IN	EP7OUT	EP6OUT	EP5OUT	EP4OUT	EP3OUT	EP2OUT	EP1OUT	Reserved						SOF	Reset	Resume	Suspend
	Read/Write									r	r	r	r	r	r	r	r	r	r	r	r	r	r	r							r	r	r	r
	Reset Value	0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	USB_INTRE	Reserved								EP7INE	EP6INE	EP5INE	EP4INE	EP3INE	EP2INE	EP1INE	EP0INE	EP7OUTE	EP6OUET	EP5OUTE	EP4OUTE	EP3OUTE	EP2OUTE	EP1OUTE	Reserved					EN_SOF	EN_Reset	EN_Resume	EN_Suspend
	Read/Write									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw	rw
	Reset Value	0								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0					0	1	1	0
0xC	USB_FRAME	Reserved												INDEX				Reserved							FramNUM								
	Read/Write													rw											r								
	Reset Value	0												0				0							0								
0x10	USB_EP0CSR	Reserved																COUNT0												ServicedSetup	ServicedOutPk	SendStall	SetupEnd	DataEnd	SentStall	InPktRdy	OutPktRdy
	Read/Write																	r												w1	w1	w1	r	w1	rw0	rw1	r
	Reset Value	0																0							0					0	0	0	0	0	0	0	0
0x14	USB_INEPxCSR	Reserved								INMAXP								Res.	ClrDataTog	SentStall	SendStall	FlushFIFO	UnderRun	FIFONotEmpty	InPktRdy	AutoSet	ISO	Mode	DMAEnab	FrcDataTog	Res.			
	Read/Write									rw									w1	rw0	rw	w1	rw0	rw0	r/set	rw	rw	rw	rw	rw				
	Reset Value	0								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x18	USB_OUTEP	Reserved								OUTMAXP								Res.	ClrDataTog	SentStall	SendStall	FlushFIFO	UnderRun	FIFONotEmpty	InPktRdy	AutoSet	ISO	Mode	DMAEnab	FrcDataTog	Res.			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	xCSR																	a	S	S	h	E	R	F	k	C		n	o				
	Read/Write									rw								w1	rw0	rw	w1	r	rw0	r	rw0	rw	rw	rw	rw				
	Reset Value	0								0								0	0	0	0	0	0	0	0	0	0	0	0	0			
0x1C	USB_OUTCOUNT	Reserved																					OUTCOUNT										
	Read/Write																						r										
	Reset Value	0																					0										
0x20 ~ 0x3F	USB_FIFODATA	FIFODATA(0~7, each EP occupies 4 Byte address)																															
	Read/Write	rw																															
	Reset Value	0																															

24. CAN bus controller

24.1. Introduction

CAN (Controller Area Network) bus is a bus standard that enables microprocessors or devices to communicate with each other without a host.

The CAN FD controller follows the CAN bus CAN2.0 (2.0A, CAN2.0B) and CAN FD protocol.

The CAN bus controller can handle the sending and receiving of data on the bus. In this product, the CAN FD controller has 12 groups of filters. The filters are used to select the messages to be received for the application.

The application in the CAN FD controller can send transmit data to the bus via one high-priority Primary Transmit Buffer (hereinafter referred to as PTB) and three Secondary Transmit Buffers (hereinafter referred to as STB), and the transmit scheduler determines the mailbox sending order. The three STBs and the three RBs can be understood as two three-level FIFOs, which are fully controlled by the hardware.

The CAN FD bus controller can also support time-trigger communication.

24.2. Main features

- Fully supports CAN2.0A/CAN2.0B/CAN FD protocols.
- CAN2.0 supports the highest communication baud rate of 1Mbit/s
- Supports baud rate pre-scaling from 1 to 1/32, flexible configuration of baud rate
- 3 receive buffers
  - FIFO method
  - Errors or non-received data do not overwrite stored messages
- 1 high priority primary transmit buffer PTB
- 3 sub-send buffers STB
  - FIFO method
  - Priority arbitration method
- 12 independent groups of filters
  - Supports 11-bit standard ID and 29-bit extended ID
  - Programmable ID CODE bits and MASK bits
- PTB/STB both support single send mode
- Silent mode support
- Supports loopback mode
- Support for capturing the type of error transmitted and locating the location of arbitration failure
- Programmable error warning values
- Support for ISO 11898-4 time triggered CAN and receive timestamps

24.3. Function Description

24.3.1. Module Block Diagram

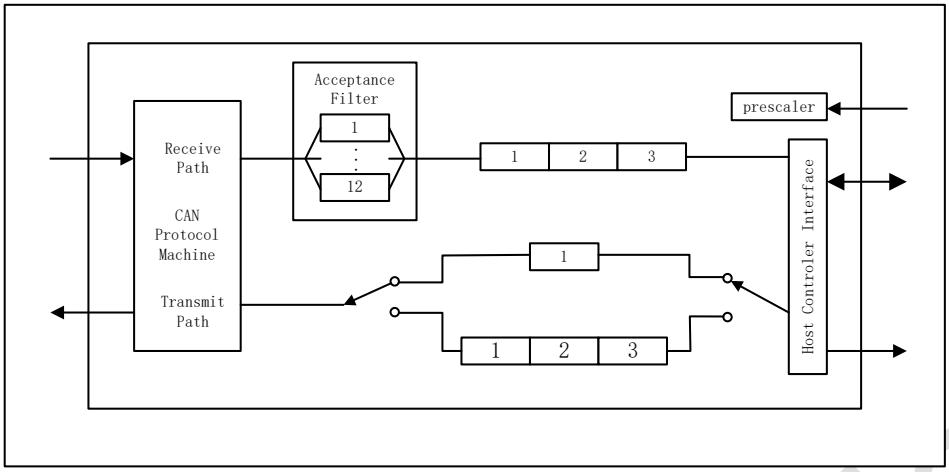


Figure24-1 CAN FD Module Block Diagram

24.3.2. Action Mode

Two operation modes exist for the CAN FD controller, reset mode (CANFD_MCR.RESET=1) and action mode (CANFD_MCR.RESET=0). When the module is initialized, you should first set the registers that can only be operated in the reset mode, then exit the reset mode and operate the rest of the registers in the action mode.

24.3.3. Baud rate setting

The CAN communication clock can_clk is based on an external high speed oscillator. Before using the CAN module, the CAN communication clock needs to be set in the RCC section.

The following figure shows the CAN bit time definition, the upper part of the dotted line is the bit time defined by the CAN protocol and the lower part of the dotted line is the bit time defined by this CAN controller CAN-CTRL. The segment1 and segment2 can be set by the registers CANFD_ACBTR and CANFD_FDBTR. The registers CANFD_ACBTR and CANFD_FDBTR can only be set when CANFD_MCR.RESET=1, i.e. CAN software reset. the CANFD_ACBTR register is used for the arbitration segment of CAN2.0 and CAN FD, and the CANFD_FDBTR register is used for the CAN FD data segment.

For FD communication, it is recommended to select 20MHz/40MHz/80MHz for the communication clock.

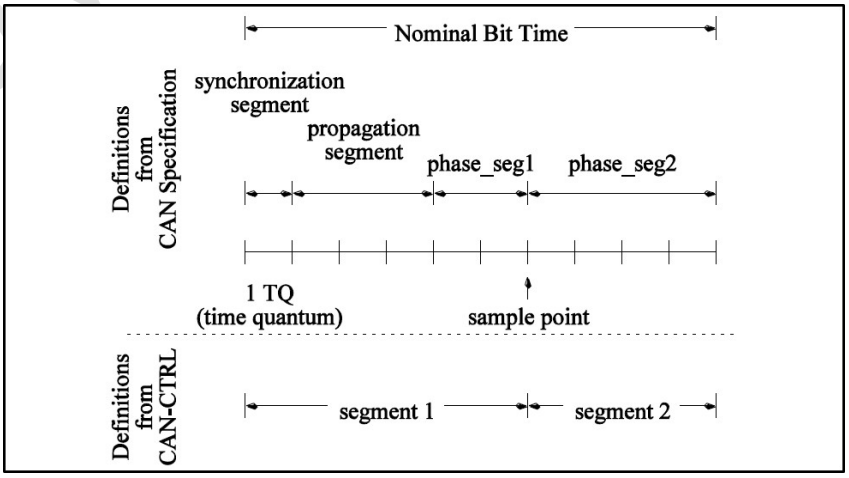


Figure24-2 CAN bit time definition

Refer to the following equation for the TQ calculation, where PRESC is set via the PRESC bit of CANFD_RLSSP. f_{can_clk} is the CAN communication clock frequency.

$$S\_TQ = \frac{S\_PRESC + 1}{fcan\_clk}$$

$$F\_TQ = \frac{F\_PRESC + 1}{fcan\_clk}$$

Please refer to the following formula for the bit time calculation, where AC_segment1 and AC_segment2 are set by AC_SEG_1 and AC_SEG_2 bits of CANFD_ACBTR register, and FD_segment1 and FD_segment2 are set by FD_SEG_1 and FD_SEG_2 bits of CANFD_FDBTR register.

Slow_BT = $t_{S\_segment1} + t_{S\_segment2}$ = ((AC_SEG_1 + 2) + (AC + SEG_2 + 1)) × AC_TQ

Fast_BT = $t_{F\_segment1} + t_{F\_segment2}$ = ((FD_SEG_1 + 2) + (FD + SEG_2 + 1)) × FD_TQ

Table 24-1 CAN time setting rule

Bit	Setting range			Rule
AC_SEG_1 bit of CANFD_ACBTR register	[0..63]	CAN2.0 bits	(slow)	SEG_1 ≥ SEG_2 + 1
	[0..63]	CAN FD nominal bits	(slow)	SEG_2 ≥ SJW
AC_SEG_2 bit of CANFD_ACBTR register	[0..7]	CAN2.0 bits	(slow)	
	[0..31]	CAN FD nominal bits	(slow)	
AC_SJW bit of CANFD_ACBTR register	[0..15]	CAN2.0 bits	(slow)	
	[0..15]	CAN FD nominal bits	(slow)	
FD_SEG_1 bit of CANFD_FDBTR register	[0..15]	CAN FD data bits	(fast)	SEG_1 ≥ SEG_2
FD_SEG_2 bit of CANFD_FDBTR register	[0..7]	CAN FD data bits	(fast)	SEG_2 ≥ SJW
FD_SJW bit of CANFD_FDBTR register	[0..7]	CAN FD data bits	(fast)	

The following are the recommended baud rate settings for CANFD for reference only.

PSP: Primary sampling point

SSP: Secondary sampling point

Seg 1: Segment 1

Seg 2: Segment 2

TDC: Transmit delay compensation

Table 24-2 Recommended baud rate setting at 20MHz communication clock

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [CAN communication clock]
0.25 (Arbitration)	80	-	1	80	64	16	16	-
0.5	80	-	1	40	32	8	8	-

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [CAN communication clock]
(Arbitration)								
0.5	80	Prohibition	1	40	32	8	8	-
1	80	80	1	20	16	4	4	16
2	80	80	1	10	8	2	2	8
4	80	80	1	5	4	1	1	4
5	75	75	1	4	3	1	1	3

Table 24-3 Recommended baud rate setting at 40MHz communication clock

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [CAN communication clock]
0.25 (Arbitration)	80	-	2	80	64	16	16	-
0.5 (Arbitration)	80	-	1	80	64	16	16	-
0.5	80	Prohibition	2	40	32	8	8	-
1	80	80	1	40	32	8	8	32
2	80	80	1	20	16	4	4	16
4	80	80	1	10	8	2	2	8
5	75	75	1	8	6	2	2	6
8	80	80	1	5	4	1	1	4

Table 24-4 Recommended baud rate setting at 80MHz communication clock

Bit Rate [Mbit/s]	PSP [%]	SSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [CAN communication clock]
0.25 (Arbitration)	80	-	4	80	64	16	16	-
0.5 (Arbitration)	80	-	2	80	64	16	16	-
0.5	80	Prohibition	4	40	32	8	8	-
1	80	80	2	40	32	8	8	64
2	80	80	2	20	16	4	4	32
4	80	80	1	20	16	4	4	16
5	75	75	1	16	12	4	4	12
8	80	80	1	10	8	2	2	8

24.3.4. Send buffer

CAN_CTRL provides two transmit buffers for sending data, the primary transmit buffer PTB and the secondary transmit buffer STB. PTB has the highest priority but can only buffer one frame of data, STB has a lower priority than PTB but can buffer 3 frames of data, and the 3 frames of data in STB can work in FIFO mode or priority arbitration mode. 3 frames of data in STB can be sent all by setting the TSALL bit of CANFD_MCR register to 1. The 3 frames of data in STB can be sent by setting the TSALL bit in CANFD_MCR register to 1. In FIFO mode, the first written data is sent first, and in priority mode, the data with smaller ID is sent first.

The data in the PTB has the highest priority, so PTB sends can defer STB sends, but STBs that have won arbitration and started sending cannot be deferred by PTB sends.

The PTB and STB can be accessed through the TBUF register. The next SLOT in the STB is selected by the TBSEL bit of the CANFD_MCR register, TBSEL=0 for PTB and TBSEL=1 for STB. The correspondence is shown in the following figure:

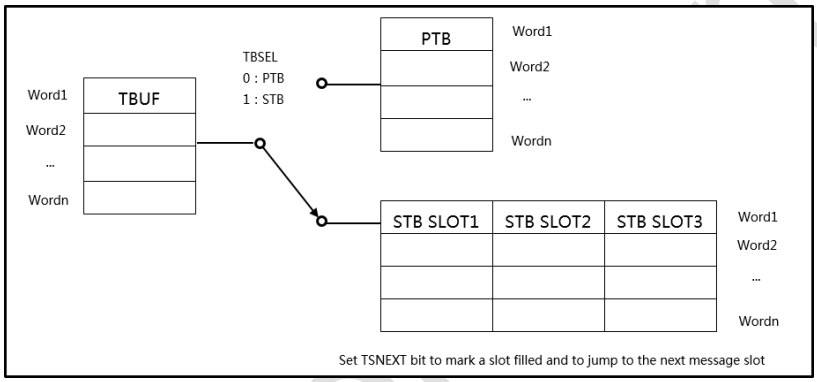


Figure 24-3 CAN FD TBUF register write send buffer and schematic

24.3.5. Receiving buffer

CAN_CTRL provides 3 SLOT receive buffers for storing the received data, the 3 SLOT receive buffers work in FIFO mode. RB SLOT reads the received data through the CANFD_RBUF register, always reads the first received data, and releases the already The RB SLOT is read and pointed to the next RB SLOT.

The diagram of RB SLOT reading by RBUF is as follows.

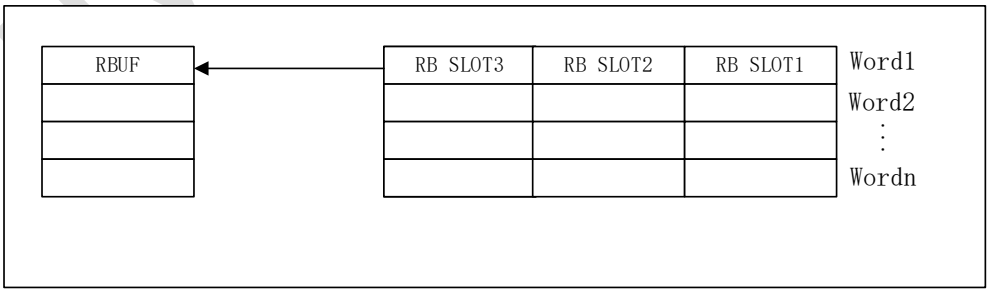


Figure 24-4 CAN RBUF register read receive buffer schematic

24.3.6. Receive filter register set

CAN_CTRL provides 12 groups of 32-bit filters for filtering the received data to reduce the CPU load, the filters can support either the standard format 11-bit ID or the extended format 29-bit ID. The ID CODE register is used to compare the received CAN IDs and the ID MASK register is used to select

the CAN ID bits for comparison. When the corresponding ID MASK bit is 1, the ID CODE of that bit is not compared.

The received data is received as long as it passes any of the 12 group filters and the received data is stored in RB, otherwise the data is not received and not stored.

The ID CODE and ID MASK are set via the CANFD_ACFCR register and the filter address is selected via the ACFADR bit of the CANFD_ACFCR register. The ID CODE and ID MASK are accessed through the CANFD_ACFC register and the CANFD_ACFM register and can only be set when CANFD_MCR.RESET=1, i.e. CAN software reset. Refer to the following figure for the ACF register access to the filter register set.

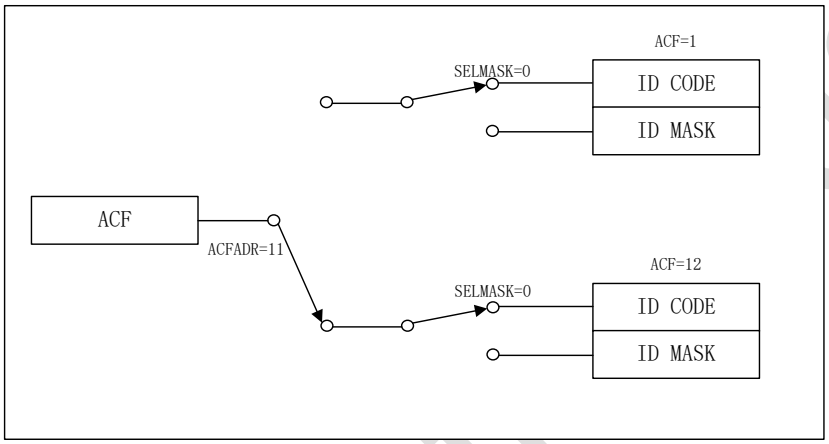


Figure 24-5 Schematic diagram of CAN ACF register access filter group

The following table shows the definition of a logical link control frame (LLC) containing a time stamp. This uniform definition is used for sending frames (stored in CANFD_TBUF), receiving frames (read from CANFD_RBUF) and configuring the receive filters (CANFD_ACFC and CANFD_ACFM).

Offset Address	Register																
0x00	CAN_ID	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Res.	Res.	Res.	ID[28:16]												
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		ID[15:0]															
0x04	CAN_FORMAT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Res.	Res.	Res.	LBF	ESI	KOER[2:0]			Res.	Res.	SEC	RMF	XLF	BRS	FDF	IDE
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Res.	Res.	Res.	Res.	Res.	DLC[10:0]										
0x0	CAN_TYPE	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	1

Offset Address	Register																
8																	6
		HANDLE[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Res.															
0x0c	CAN_AF	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		AF[31:16]															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		AF[15:0]															
0x18	CAN_TTCAN	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		CYCLE_TIME[15:0]															
0x1c	CAN_DATA1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Data[31:16]															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Data[15:0]															
……																	
0x20	CAN_DATA16	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Data[31:16]															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Data[15:0]															

Bit	Description
ID	Frame identifier
DLC	Data Length Code The DLC defines the number of bytes of payload within the frame (Table 3-42). The DLC length for classic CAN 2.0B frames is 4 bits (DLC(3:0)). For classic CAN 2.0B, it is optional to transmit remote frames where the DLC is not meaningful. The explanation of the bit RMF provides more details.
IDE	Identifier Extension 0 – Standard Format: ID(28:18) 1 – Extended Format: ID(28:0)
FDF	This bit should be set to 0 to support CAN 2.0 frame
BRS	CAN FD Bit Rate Switch Enable

Bit	Description
	0 - nominal/slow bit rate for full frames 1 - Switch to data/fast bit rate for data payload and CRC of CAN FD frames BRS is not meaningful for classic CAN.
RMF	Remote frames 0 - Data frames 1 - Remote frame Remote frames carry a zero-byte payload. The DLC value is transmitted as is, but has no effect on the frame size. Therefore, the DLC value of a remote frame may carry some encoding information. Only classic CAN 2.0B frames can be remote frames.
KOER	Error type The KOER of the received frame has the same meaning as the KOER bit in register EALCAP. If RBALL=1 (chapter 4.10.4), the KOER of the received frame becomes meaningful. Note that if RBALL=1, the filter set is normally disabled.
ESI	Error Status Indicator The protocol machine automatically embeds the correct value of the ESI into the transmitted frame. 0 - CAN node in error state 1 - CAN node error passive The ESI is only included in the CAN FD frame, so it is always 0 for the received classic CAN 2.0B. The error status of the transmission is indicated by the bit EPASS in the register INTF.
LBF	Loopback Frames If loopback mode is activated and the CAN controller has received its own transmission frame, the LBF of the received frame is set to 1. This is useful if LBME=1 and other nodes in the network are also transmitting.
HANDLE	Frame identification HANDLE The purpose of the handle is to identify the frame using TSTAT. HANDLE is not used in MAC frames. It is recommended that the host application write the value of the software counter to HANDLE. Such a software counter can be incremented with each new frame to be transmitted, and should be flipped. Note: To avoid uninitialized memory problems during emulation, the host application should always define HANDLE for each frame to be transmitted.
CYCLE_TIME	CYCLE_TIME (timestamp of the TTCAN) CYCLE_TIME will store only the received frames. This is the cycle time at the SOF of the frame. The cycle time of the reference message is always 0.
AF	Accept field Recommended to be used mainly for frame acceptance filtering, with all reserved bits written oxFFFFFFFF.
Payload Data	The payload data of the frame. Up to 8 bytes for Classic CAN 2.0B and up to 64 bytes for CANFD.

24.3.7. Data transmission

At least one frame of data must be loaded in the PTB or STB before starting the transmission, the TPE is locked during the PTB transmission, and the STB filling can be confirmed by the TSSTAT bit of the CANFD_MCR register. The transmission data setting procedure is as follows:

- Set TBSEL to choose from PTB and STB to send BUF
- Write the data to be sent through CANFD_TBUF register.
- If STB is selected, set TSNEXT=1 to complete the loading of STB SLOT.
- Transmit Enable
  - PTB send using TPE
  - STB send using TSALL or TSONE
- Send completion status confirmation
  - PTB use TPIF for send completion, TPIE for enabling TPIF
  - STB uses TSONE to use TSIF when the transmission is completed, and TSIE is used to enable TSIF
  - TSIF is used when the STB uses TSALL to finish sending, and TSIF will be set after all STB SLOT data is sent, and TSIE is used to enable TSIF.

24.3.8. Single data transmission

The TPE bit of the CANFD_MCR register is used to set the single send mode for PTB and the TSONE bit is used to set the single send mode for STB when the automatic retransmission function is not required. When the data is successfully sent, the single send operation is the same as the normal send mode. However, the following results occur when the data is not sent successfully:

- TPIF is set (TPIE=1) and the corresponding BUF SLOT data is cleared.
- When there is an error sent, KOER is updated and BEIF is set (BEIE=1).
- Arbitration fails, ALIF is set (ALIE=1).
- In single send mode, you cannot rely on TPIF alone to determine whether the send is complete, but need to determine whether the send is complete together with BEIF and ALIF.

24.3.9. Cancel data sending

A data send that has been requested but not yet executed can be canceled via the TPA or TSA of the CANFD_MCR register. Cancellation of a data send can occur in the following cases:

- Arbitration in progress
  - If node arbitration fails, the data transmission is cancelled.
  - If node arbitration succeeds, the transmission continues.
- Data transmission in progress
  - The data is successfully sent and ACK is received, and the corresponding flag and status are set normally. Data sending is not canceled.
  - Successful sending of data but no ACK received, data sending is cancelled and error counter is increased.
  - TSALL=1 Set to send data, STB SLOT data being sent is sent normally, and STB SLOT that did not start sending is cancelled.
- The cancellation of data sending results in the following two cases.
  - The TPA releases the PTB and makes TPE=0.

- TSA releases one STB SLOT or all STB SLOTs depending on whether it is a TSONE or TSALL enabled transmission.

24.3.10. Data reception

The receive filter group reduces the CPU load by filtering out unwanted receive data and reducing the occurrence of interrupts and RB reads. The receive data setting procedure is as follows:

1) Set the filter group.
2) Set RFIE, RAFIE and AFWL.
3) Wait for RFIF or RAFIF.
4) Read the earliest received data from RB FIFO via RBUF.
5) Set RREL=1 to select the next RB SLOT.
6) Repeat 4, 5 until RB is confirmed empty by RSTAT.

24.3.11. Error Handling

CAN_CTRL can on the one hand handle some errors automatically, e.g. by automatically resending data or discarding frames containing errors, and on the other hand report errors to the CPU via interrupts.

The CAN node has the following three error states:

- Error active: The node automatically sends active error flags when it detects an error.
- Error Passive: The node automatically sends a passive error flag when it detects an error.
- Node Off: In the off state this node no longer affects the whole CAN network.

CAN_CTRL provides the TECNT and RECNT counters of the CANFD_WECR register for error counting; the TECNT and RECNT counters are incremented and decremented according to the rules specified in the CAN protocol. In addition, the EWL bit of the programmable CAN error warning CANFD_WECR register is provided to generate an error interrupt to notify the CPU.

There are five types of errors during CAN communication, which can be identified by the KOER bit of the EALCAP register.

- Bit Error
- Form error
- Fill error
- Response error
- CRC error

24.3.12. Node closure

When the number of transmit errors is greater than 255, the CAN node automatically enters the Node Off state and does not participate in CAN communication until it returns to the Error Active state. The CAN node shutdown state can be confirmed by the BUSOFF bit in the CANFD_MCR register, and the EIF interrupt in the CANFD_IFR register is generated when BUSOFF is set.

There are two ways to restore the CAN from the Node Off state to the Error Active state:

- Power-on reset
- Receive 128 consecutive 11-bit invisible bit sequences (recovery sequences)

In the node off state, the TECNT value remains unchanged and RECNT is used to count the recovery sequence. After recovering from the node shutdown state, TECNT and RECNT are reset to 0.

24.3.13. Arbitration Failure Location Capture

CAN_CTRL captures the exact location of the arbitration failure bit and reflects it in the ALC register of the CANFD_WECR register. the ALC register holds the location of the most recent arbitration failure bit, and the ALC bit is not updated if the node wins the arbitration. the ALC value is defined as follows:

After the SOF bit, the first ID data bit ALC is 0, the second ID data bit ALC is 1, and so on. Since arbitration occurs only within the arbitration field, the maximum value of ALC is 31. For example, if a standard format remote frame and an extended frame arbitrate, and the extended frame fails at the IDE bit, ALC = 12.

24.3.14. Loopback mode

CAN_CTRL support the following two loopback modes:

- Internal Loopback
- External Loopback

Both loopback modes receive their own outgoing data frames and are mainly used for testing purposes.

In internal loopback mode, the module internally connects the receive data line to the transmit data line and the transmit data is not output. In internal loopback mode, the node generates a self-answer signal to avoid ACK errors.

The external loopback mode maintains the connection to the transceiver so that the sent data is still present on the CAN bus and the CAN receives its own sent data with the help of the transceiver. The external loopback mode can be determined by the SACK bit in the CANFD_MCR register to generate a self-response signal or not.

The external loopback mode, with SACK=0, results in the following two cases:

- Other nodes also receive the data frame sent by this node and send an answer signal, in this case this node is able to send and receive data successfully.
- If no other node returns an answer signal, an answer error is generated and the data is resent and the error counter is increased. In this case, it is recommended to use single send mode.

When returning from loopback mode to normal mode, a software reset of CAN_CTRL is required in addition to clearing the mode bit.

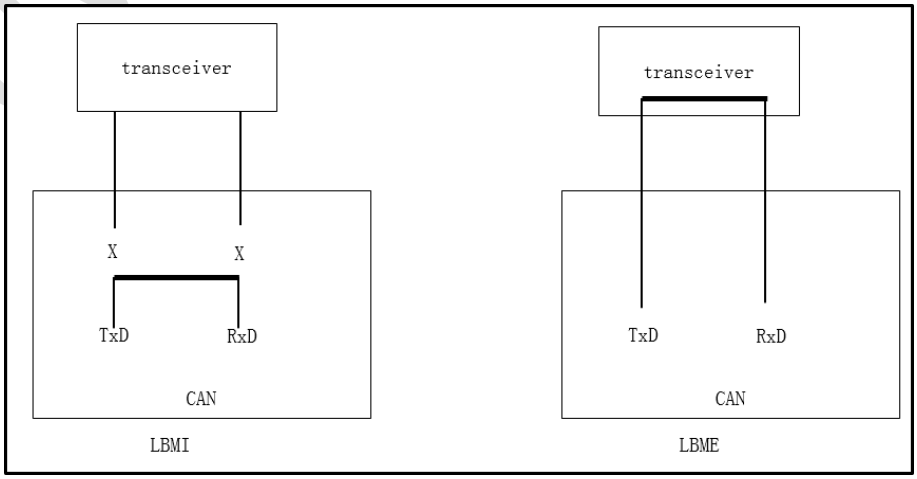


Figure 24-6 CANFD Internal loopback LBMI and external loopback LBME diagram

24.3.15. Silent mode

Silent mode can be used to monitor CAN network data. In silent mode, data can be received from the CAN bus and no data is sent to the bus. Set the LOM in the CANFD_MCR register to 1 to put the CAN bus controller into silent mode and clear it to 0 to leave silent mode.

The external loopback mode can be combined with the silent mode to form the external loopback silent mode, where the CAN can be considered a quiet receiver, but can send data when necessary. In external loopback silent mode, frames containing self-answering signals are allowed to be sent, but the node does not generate error flags and overloaded frames.

24.3.16. Software reset function

The software reset function is implemented by setting the RESET bit of CANFD_MCR register to 1. The set range of the software reset function is shown in the table below.

Table 24-5 Software Reset Range Table

Register bit name	Software Reset	Remark	Register bit name	Software Reset	Remark
ACFADR	NO	-	FD_SEG_1	Yes	Writable only during software reset
ACODE	NO	Writable only during software reset	FD_SEG_2	Yes	Writable only during software reset
AE_x	NO	-	FD_SJW	Yes	Writable only during software reset
AFWL	NO	-	KOER	Yes	-
AIF	Yes	-	LBME	Yes	-
ALC	Yes	-	LBMI	Yes	-
ALIE	NO	-	RACTIVE	Yes	Receiving stops immediately and no ACK is generated
ALIF	Yes	-	RAFIE	NO	-
AMASK	NO	Writable only during software reset	RAFIF	Yes	-
BEIE	NO	-	RBALL	Yes	-
BEIF	Yes	-	CANFD_RBUF	Yes	RB is marked as empty with variable value
BUSOFF	NO	Clear by writing 1	RECNT	NO	Clear zero by BUSOFF write 1
EIE	NO	-	REF_ID	NO	-
EIF	NO	-	REF_IDE	NO	-
EPASS	NO	-	RFIE	NO	-
EPIE	NO	-	RFIF	Yes	-
EPIF	Yes	-	RIE	NO	-
EWARN	NO	-	RIF	Yes	-
EWL	Yes	-	ROIE	NO	-
FD_ISO	NO	Writable only during software reset	ROIF	Yes	-
			ROM	NO	
ROV	Yes	-	TSMODE	NO	

Register bit name	Software Reset	Remark	Register bit name	Software Reset	Remark
RREL	Yes	-	TSNEXT	Yes	-
RSTAT	Yes		TSONE	Yes	-
SACK	Yes	-	TPIE	NO	-
SELMASK	NO	-	TPIF	Yes	-
PRESC	NO	Writable only during software reset	TPSS	Yes	-
AC_SEG_1	NO	Writable only during software reset	TSFF	Yes	All STB SLOTs are marked as empty
AC_SEG_2	NO	Writable only during software reset	TSIE	NO	-
AC_SJW	NO	Writable only during software reset	TSIF	Yes	-
SSPOFF	Yes	-	TSSS	Yes	-
TACTIVE	Yes	Sending stops immediately	TSSTAT	Yes	All STB SLOTs are marked as empty
TBE	Yes	-	TTEN	Yes	-
TBF	NO	-	TTIF	Yes	-
TBPTR	NO	-	TTIE	NO	-
TBSEL	Yes	-	TTPTR	NO	-
TBUF	Yes	STB is marked as empty and points to PTB	TTTBM	NO	-
TDCEN	Yes	-	TTYPE	NO	-
TECNT	NO	Can be cleared by BUSOFF=1	TT_TRIG	NO	-
TEIF	Yes	-	TT_WTRIG	NO	-
TPA	Yes	-	T_PRESC	NO	-
TPE	Yes	-	WTIE	NO	-
TSA	Yes	-	WTIF	Yes	
TSALL	Yes	-			

24.3.17. Compatible with CAN-FD function

CAN-CTRL When the CAN FD function is disabled, even if CAN FD frames are received in a network containing CAN FD, the receiver automatically ignores these frames, does not return the ACK and waits until the bus is free to send or receive the next CAN2.0B frame.

24.3.18. Time-triggered TTCAN

CAN-CTRL provides partial (lever 1) hardware support for the time-triggered communication method specified in ISO 11898-4. This section describes the TTCAN functionality in the following 5 sections.

24.3.18.1. TBUF behavior in TTCAN mode

TTTBM=1

When TTTBM=1, the PTB and STB SLOT form the same TB SLOT, and the transmit BUF is specified by the TBPTR bit in the CANFD_TTCR register, where TBPTR=0 points to the PTB,

TBPTR=1 points to the STB SLOT1, and so on. The host can use the TBE and TBF bits of the CANFD_TTCR register to mark the transmit BUF SLOT, where the TBSEL and TSNEXT registers have no meaning and can be ignored.

When TTTBM=1, the PTB does not have any special attributes and, like the STB SLOT, the Transmission Complete flag uses the TSIF bit of the CANFD_IFR register.

In TTCAN mode, the transmit BUF has no FIFO mode and no priority arbitration mode, and only one selected SLOT can send data.

In TTCAN mode, the start of transmission is time-triggered and TPE, TSONE, TSALL, and TPA are fixed to 0 and are ignored.

TTTBM=0

When TTTBM=0, a combination of event-driven communication and receive timestamp functions are used. In this mode, the functions of PTB and STB are the same as when TTEN=0, so PTB always has the highest priority, while STB can work in FIFO mode or arbitration mode.

24.3.18.2. TTCAN Feature

After power-up, the Time Master needs to be initialized according to the ISO 11898-4 protocol. There can be up to 8 potential Time Mas-ter in a CAN network, each Time Master has its own reference message ID (last 3 bits of the ID). After TTEN=1, the 16-bit counter starts working and when the reference message is successfully received or the Time Master successfully sends a reference message, the CAN controller copies the Sync_Mark to the Ref_Mark, which sets the cycle time to 0. The successful reception of the reference message sets the RIF flag and the successful sending of the reference message sets the TPIF or TSIF flag. At this point the host needs to prepare the trigger condition for the next action.

The trigger condition can be a receive trigger. This trigger only triggers an interrupt that can be used to detect that an expected message has not been received. The trigger condition can also be a send trigger. This trigger starts sending the data in the TBUF SLOT specified by the TTPTR bit of the CANFD_TTCR register. If the selected TBUF SLOT is marked as empty, sending does not start, but the interrupt flag is set.

24.3.18.3. TTCAN Timing

CAN_CTRL supports ISO11898-4 level 1. contains a 16-bit counter operating at the bit times defined by AC_PRESC, AC_SEG_1, AC_SEG_2. If TTEN=1, there is an additional prescaler T_PRESC.

The value of the counter at SOF of a frame of data is Sync_Mark. if the frame of data is a reference message, Sync_Mark is copied to Ref_Mark. cycle time is equal to the value of the counter minus Ref_Mark. this time is used as a timestamp for the received message or as a trigger time reference for the sent message.

24.3.18.4. TTCAN trigger method

The trigger method of the TTCAN is defined by the TTYPE bit of the CANFD_TTCR register, the TTPTR register specifies the send SLOT, and TT_TRIG specifies the cycle time of the trigger. The following five trigger methods are included:

- Immediate Trigger
- Time trigger

- Single send trigger
- Send start trigger
- Send Stop Trigger

All triggers use the TTIF flag except for the immediate trigger method. with TTTBM=1, only the time trigger method is supported.

Immediate trigger

The trigger is started by writing the high bit of TT_TRIG (not caring about the value written). In this mode, the data in the TBUF SLOT selected by TTPTR is sent immediately.TTIF is not set.

Time Trigger

The time-triggered method generates interrupts only by setting the TTIF flag and has no other function. If a node expects to receive the expected data within a specific time window, the time-triggered method can be used. If the TT_TRIG value is less than the actual cycle time, then TEIF is set and no other action is taken.

Single send trigger

The single send trigger method is used to send data within the execution time window. In this case, the RELIM bit of the CANFD_RLSSP register is ignored.

If the data does not start sending within the specified send enable time window, the frame is discarded. The corresponding transmit BUF SLOT is marked empty and the AIF is set. The data in the corresponding transmit BUF is not rewritten because it can be sent again by setting the TPF.

If the TT_TRIG value is less than the actual cycle time, TEIF is set and no other action is taken.

Send start trigger

The send start trigger method is used within the arbitration time window to participate in arbitration. RELIM is used to determine whether to automatically retransmit or single send mode. If the message specified in the TTPTR register is not successfully sent, the send stop trigger can be used to stop the send.

If the TT_TRIG value is less than the actual cycle time, TEIF is set and no other action is taken.

Send Stop Trigger

The transmit stop trigger is used to stop a transmission that has been started by the transmit start trigger. If the transmission is stopped, the transmission frame is discarded, AIF is set and the selected TBUF SLOT is marked empty, but the data in the TBUF SLOT is not rewritten and can be sent again by setting the TPF.

If the TT_TRIG value is less than the actual cycle time then TEIF is set and execution stops.

24.3.18.5. TTCAN trigger watch time

The TTCAN trigger watchdog time function is similar to the watchdog function and is used when TTTBM=1. It is used to see how long the watchdog has been in operation since the last successful reception of a reference message. The reference message can be received during the cycle time or after an event, and the application should set the appropriate watchdog time for each case.

If cycle count is equal to TT_WTRIG, WTIF is set. write 0 via WTIE to turn off the watchdog trigger. If TT_WTRIG is smaller than the actual cycle time, then TEIF is set.

24.3.19. TDC or RDC

TDC (Transmitter Delay Compensation) and RDC (Receiver Delay Compensation) are used to compensate the data delay when CANFD communication may exceed one bit time. TDC (Transmitter Delay Compensation) and RDC (Receiver Delay Compensation) are used to compensate for data delay, where TDC requires a software-controlled switch, while RDC is not required and is effective automatically. The following figure:

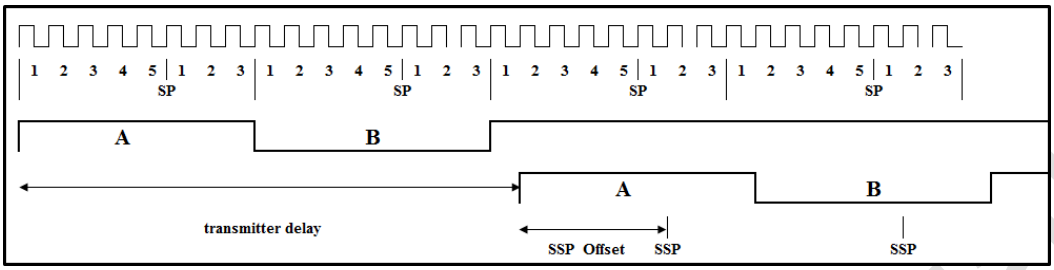


Figure 24-7 TDC function diagram

When the software enables the TDC function, this controller can automatically determine the transmitter delay and set the SSP Offset by setting the SSPOFF in the CANFD_RLSSP register. the recommended SSP Offset setting value is the same as $tF\_s\ egmeent1$ setting value.

24.3.20. Interruption

Symbol	Interrupt flag	Description
CAN_HOST	RIF	Receive interrupt
	ROIF	Receive overflow interrupt
	ROIF	Receive BUF full interrupt
	RAFIF	Receive BUF will be full interrupt
	TPIF	PTB transmit interrupt
	TSIF	STB send interrupt
	EIF	Error interruption
	AIF	Cancel send interrupt
	EPIE	Error Passive Interrupt
	ALIF	Failed arbitration interrupt
	BEIF	Bus error interrupt
	WTIF	Trigger watchdog interrupt
	TEIF	Trigger an error interrupt
	TTIF	Time-triggered interrupt

24.4. Register Description

24.4.1. Node Configuration Register (CANFD_TSNCR)

Address offset：0x00

Reset value：0x02010801

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROP	CES
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

VERSION[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Name	R/W	Reset Value	Function
25	Res	-	-	-
24	Res	-	-	-
18	Res	-	-	-
17	ROP	rw	0	Restrict operation 0 - Restricted operation is not enabled 1 - Restricted operation enabled The ROP cannot be changed while the transmission is active. If the ROP is enabled, the transmission cannot be started.
16	CES	rw	1	CAN Error Signals 0-Disable error signal 1-Enables the error signal If CES=1, an error is signaled using the error flag and the error counter is incremented. This behavior is equivalent to the definition in ISO 11898-1:2015 (classic CAN and CAN FD). Otherwise, if CES=0, the error causes a protocol exception event and does not modify the error counter.
15:0	Version[15:0]	r		CAN-CTRL version. VER_1 holds the major version and VER_0 holds the minor version. For example, version 5x15N00S00 is represented by VER_1=5 and VER_0=15=0x0f.

24.4.2. Bit Timing Configuration Register (CANFD_ACBTR)

Address offset：0x04

Reset value：0x05050008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AC_SJW[6:0]							Res.	AC_SEG_2[6:0]						
	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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	AC_SEG_1[8:0]								
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	Name	R/W	Reset Value	Function
30:24	AC_SJW	rw	0x5	Synchronous jump width

Bit	Name	R/W	Reset Value	Function
				The sync jump width $tSJW = (x\_SJW + 1) \cdot TQ$ is the maximum time to shorten or lengthen the resynchronization bit time by the maximum time.
22:16	AC_SEG_2	rw	0x5	Bit Timing Segment 2 Time $tSEG\_2 = (x\_SEG\_2 + 1) \cdot TQ$ from the end of the sample point in place
8:0	AC_SEG_1	rw	0x8	Bit Timing Segment 1 The sampling point will be set after the start of the bit time $tSEG\_1 = (x\_SEG\_1 + 2) \cdot T\ Q$.

24.4.3. CANFD_FDBTR

Address offset：0x08

Reset value：0x02020003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FD_SJW[6:0]							Res.	FD_SEG_2[6:0]						
	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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FD_SEG_1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
30:24	FD_SJW	rw	0x2	Synchronization Jump Width The Synchronization Jump Width $tSJW = (x\_SJW + 1) \cdot TQ$ is the maximum time for shortening or lengthening the Bit Time for resynchronization.
22:16	FD_SEG_2	rw	0x2	Bit Timing Segment 2 Time $tSEG\_2 = (x\_SEG\_2 + 1) \cdot TQ$ after the sample point to the end of the bit.
7:0	FD_SEG_1	rw	0x3	Bit Timing Segment 1 The sample point will be set to $tSEG\_1 = (x\_SEG\_1 + 2) \cdot TQ$ after start of bit time.

24.4.4. Limit and Prescaler Configuration Register (CANFD_RLSSP)

Address offset：0x10

Reset value：0x77000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RETLIM[2:0]			Res.	REALIM[2:0]			Res							

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

Bit	Name	R/W	Reset Value	Function
30:28	RETLIM	rw	0x111	111 - no limit (limited only by sending error counter TECNT) 110 - 7 attempts ... 000 - 1 attempt (no retransmission) If any errors occur during transmission, the CAN node can automatically retry when the bus is idle. The number of retransmission attempts can be limited using RETLIM. The RETLIM can be updated at any time, but after the update, the contents of the registers need to be cross-transferred to the CAN protocol machine using the clock domain. For this short period of time (a few clocks), RETLIM is write-locked.
26:24	REALIM	rw	0x111	0x111 Re-arbitration limit 111 - no limit 110 - 7 attempts ... 000 - 1 attempt (no re-arbitration) If two or more CAN nodes try to transmit frames at the same time, the lower priority frame loses arbitration and silently backs off without interrupting the higher priority frame. CAN nodes can automatically retry when the bus is idle. Use REALIM to limit the number of re-arbitration attempts. RELIM can be updated at any time, but after the update, the contents of the registers need to be cross-transferred to the CAN protocol machine using the clock domain. For this short period of time (a few clocks), REALIM is write-locked.
23:16	Res	-	-	-
15:8	FD_SSPOFF	rw	0x0	Secondary Sample Point Offset If SSPOFF≠0 and CES=1, transmitter delay compensation (TDC) is enabled. If BRS is active, TDC will be activated during the data phase of a CAN FD frame or during the data phase of a CAN XL frame. For more details on TDC, see the function description. The transmitter delay plus SSPOFF defines the timing of the secondary sampling point of the TDC. SSPOFF is given in terms of the number of TQs. For CAN FD frames, FD_SSPOFF defines SSPOFF.
4:0	PRESC	rw	0x0	Prescaler

Bit	Name	R/W	Reset Value	Function
				The prescaler divides the system clock to synthesize the time quantum TQ. $nprescaler$ = $PRESC$ + 1 and tTQ = $npre\ scaler$ / $fcan\_cl\ k$

24.4.5. Status Register (CANFD_IFR)

Address offset：0x14

Reset value：0x0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EWARN	EPASS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WTIF	TEIF	TTIF	EPIF	ALIF	BEIF	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31	EWARN	r	0x0	Error warning limit reached 0: RECNT or TECNT is less than the EWL setting 1: RECNT or TECNT is greater than or equal to the EWL setting
30	EPASS	r	0x0	Error Passive mode active 0: The node is an active error node 1: Node is passive error node
18	Res	-	-	-
17	Res	-	-	-
16	Res	-	-	-
15	Res	-	-	-
14	Res	-	-	-
13	WTIF	rw	0x0	TTCAN: Watch Trigger Interrupt Flag If the cycle count reaches the limit defined by TT_WTRIG and WTIE is set, WTIF will be set.
12	TEIF	rw	0x0	TTCAN: Trigger Error Interrupt Flag The conditions for setting TEIF are described in the TTCAN section; there are no bits to enable or disable TEIF processing
11	TTIF	rw	0x0	TTCAN: Time-triggered interrupt flag If TTIE is set and the cycle time is equal to the trigger time TT_TRIG, TTIF will be set.

Bit	Name	R/W	Reset Value	Function
				Writing a 1 to TTIF will reset it. Writing a zero has no effect. TTIF will be set only once. If TT_TRIG is not updated, TTIF will not be set again in the next basic cycle.
10	EPIF	rw	0x0	Error Passive Interrupt Flag. EPIF will be activated if EPASS changes and EPIE=1.
9	ALIF	rw	0x0	Arbitration Lost Interrupt Flag 0: Arbitration successful 1: Arbitration failed Clear 0 by writing 1 to the application.
8	BEIF	rw	0x0	Bus Error Interrupt Flag 0: No bus error 1: Bus error Clear 0 by writing 1 through the application.
7	RIF	rw	0x0	Receive Interrupt Flag 0: No data frame received 1: Received a valid data frame or remote frame Clear 0 by writing 1 to the application.
6	ROIF	rw	0x0	Receive Overrun Interrupt Flag 0: No RBs are overwritten (overwrite) 1: At least one RB is overwritten ROIF and RFIF are set to 1 at the same time when the overflow occurs. 0 is cleared by writing 1 to the application.
5	RFIF	rw	0x0	Receive BUF full interrupt flag 0: RB FIFO is not full 1: RB FIFO full Clear 0 by application write 1.
4	RAFIF	rw	0x0	Receive BUF almost full interrupt flag 0: The number of filled RB SLOTs is less than the AFWL setting 1: The number of filled RB SLOTs is greater than or equal to the AFWL setting Clear 0 by writing 1 to the application.
3	TPIF	rw	0x0	Transmission Primary Interrupt Flag 0: No PTB transmission completed 1: The requested PTB transmission is successfully completed by clearing 0 by writing 1 to the application. Note: TPIF is invalid in TTCAN mode, only TSIF flag is applicable
2	TSIF	rw	0x0	Transmission Secondary Interrupt Flag 0: No STB transmission completed 1: The requested STB transmission is successfully completed by clearing 0 by writing 1 to the application.

Bit	Name	R/W	Reset Value	Function
				Note: In TTCAN mode, TPIF is invalid, only TSIF flag is used
1	EIF	rw	0x0	Error Interrupt Flag 0: The BUSOFF bit has not changed, or the value of the error counter has not changed in relation to the ERROR warning limit setting. 1: The BUSOFF bit is changed, or the relationship between the value of the error counter and the ERROR warning limit value is changed. For example, the value of the error counter changes from less than the set value to more than the set value, or from more than the set value to less than the set value. Clear 0 by writing 1 through the application.
0	AIF	rw	0x0	Abort Interrupt Flag 0: Sending data is not canceled 1: The send message requested via TPA and TSA was successfully cancelled. Clear 0 by application write 1.

24.4.6. Interrupt Enable Register (CANFD_IER)

Address offset：0x18

Reset value：0x000468fe

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WTIE	Res.	TTIE	EPIE	ALIE	BEIE	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	Res.
		rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Name	R/W	Reset Value	Function
18	Res	-	-	-
14	Res	-	-	-
13	WTIE	rw	0x1	Watch Trigger Interrupt Enable 0: Disable 1: Enable
11	TTIE	rw	0x1	Time Trigger Interrupt Enable 0: Disable 1: Enable
10	EPIE	rw	0x1	Error WARNing limit reached 0: RECNT or TECNT is less than the EWL setting

Bit	Name	R/W	Reset Value	Function
				1: RECNT or TECNT is greater than or equal to the EWL setting
9	ALIE	rw	0x1	Arbitration Lost Interrupt Enable 0: Disable 1: Enabled
8	BEIE	rw	0x1	Bus Error Interrupt Enable 0: Disable 1: Enabled
7	RIE	rw	0x1	Receive Interrupt Enable 0: Disable 1: Enable
6	ROIE	rw	0x1	Receive Overrun Interrupt Enable 0: Disable 1: Enable
5	RFIE	rw	0x1	RB Full Interrupt Enable 0: Disable 1: Enable
4	RAFIE	rw	0x1	RB Almost Full Interrupt Enable 0: Disable 1: Enable
3	TPIE	rw	0x1	Transmission Primary Interrupt Enable 0: Disable 1: Enable
2	TSIE	rw	0x1	Transmission Secondary Interrupt Enable 0: Disable 1: Enable
1	EIE	rw	0x1	Error Interrupt Enable 0: Disable 1: Enable

24.4.7. Transmission Status Register (CANFD_TSR)

Address offset：0x1c

Reset value：0x0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TSTAT_H[2:0]			HANDLE_H[7:0]							
					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
Res.	Res.	Res.	Res.	Res.	TSTAT_L[2:0]			HANDLE_L[7:0]							
					r	r	r	r	r	r	r	r	r	r	r

Bit	Name	R/W	Reset Value	Function
26:24	TSTAT_H	rw	0	Transmission Status Code

23:16	HANDLE_H	rw	0	Frame Recognition Handle
10:8	TSTAT_L	rw	0	Transmission Status Code
7:0	HANDLE_L	rw	0	Frame Recognition Handle

24.4.8. Global Configuration Register (CANFD_MCR)

Address offset：0x28

Reset value：0x00900080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SACK	ROM	ROV	RREL	RBALL	Res.	RSTAT[1:0]		FD_ISO	TSNEXT	TSMODE	TTTBM	Res.	TSFF	TSSTAT[1:0]	
rw	rw	r	rw	rw		r	r	rw	rw	rw	rw		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	RESET	LBME	LBMI	Res.	Res.	Res.	Res.	BUSOFF
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					rw

Bit	Name	R/W	Reset Value	Function
31	SACK	rw	0	Self-Acknowledge 0: No self-acknowledge 1: When LBME=1, enable self-ACKnowledge function
30	ROM	rw	0	Receive buffer Overflow Mode 0: The earliest received data is overwritten 1: The newly received data is not stored
29	ROV	r	0	Receive buffer OVerflow 0: No overflow 1: Overflow, at least one data loss Cleared by writing RREL to 1.
28	RREL	rw	0	Receive buffer RELease 0: not released 1: means the receive BUF has been read and the RBUF register points to the next RB SLOT.
27	RBALL	rw	0	Receive Buffer stores ALL data frames 0: normal mode 1: stores all data including those with errors.
25:24	RSTAT	r	0	Receive buffer status 00 - Empty 01 -> empty and < almost full (AFWL)

Bit	Name	R/W	Reset Value	Function
				10 - Almost full (programmable threshold for AFWL) but not full and no overflow 11 - full (hold setting on overflow - see ROV for overflow signal)
23	FD_ISO	rw	1	CAN FD ISO mode 0 - Bosch CAN FD (non-ISO) mode 1 - ISO CAN FD mode (ISO 11898-1:2015) The ISO CAN FD mode has a different CRC initialization value and an additional fill bit count. The two modes are not compatible and must not be mixed in a CAN network. This bit has no effect on CAN 2.0B. This bit is only writable when RESET=1.
22	TSNEXT	rw	0	Transmit buffer Secondary NEXT 0: No action 1: Current STB SLOT is filled, pointing to the next SLOT After the application writes the data in the TBUF, the application identifies that the current STB SLOT is filled by setting the TSNEXT bit, so that the hardware can point the TBUF to the next STB SLOT. The data in the STB SLOT identified by the TSNEXT bit can be sent via the TSONE or TSALL bit. This bit is cleared to zero by hardware through an application write 1. After all STB SLOTs are filled, TSNEXT is held at 1 until an STB SLOT is released. Note: This bit is fixed to 0 in TTCAN mode.
21	TSMODE	rw	0	Transmit buffer Secondary operation MODE 0: FIFO mode 1: Priority mode FIFO mode sends data frames in the order they are written. The priority mode is automatically determined by the ID, the smaller the ID, the higher the priority. The PTB has the highest priority regardless of the mode. Note: TSMODE bit can only be set when STB is empty.
20	TTTBM	rw	1	TTCAN Transmit Buffer Mode TTTBM is ignored when TTEN=0. 0: determined by TSMODE, PTB and STB 1: Set by TBPTR and TTPTR This bit can be set to 0 when only the receive timestamp function is required in TTCAN mode, and the use of PTB or STB is determined by TSMODE. Note: TSMODE bit can only be set when STB is empty.
18	TSFF	r	0	TTEN=0 or TTTBM=0: Transmit Secondary buffer Full Flag

Bit	Name	R/W	Reset Value	Function
				0: STB SLOT is not fully filled 1: STB SLOT is fully filled TTEN=1 and TTTBM=1: Transmit buffer Full Flag 0: The transmit BUF selected by TBPTR is not fully filled 1: The transmit BUF selected by TBPTR is fully filled
17:16	TSSTAT	r	0	Transmission Secondary Status bits TTEN=0 or TTEN=1 &TTTBM=0 00: STB empty 01: STB less than or equal to half full 10: STB greater than half full 11: STB full TTEN=1 and TTTBM=1 00：PTB and STB empty 01: PTB and STB not full 10: Reserved 11: PTB and STB full
15	TBSEL	rw	0	Transmit Buffer Select 0：PTB 1: STB When TTEN=1 & TTTBM=1, TBSEL is reset to the reset value. Note: When writing the TBUF register or TSNEXT bit, this bit needs to keep the fixed value.
14	LOM	rw	0	Silent mode enable bit (Listen Only Mode) 0: Silent mode is disabled 1: Enable silent mode LOM=1&LBME=0 to disable transmitting. When LOM=1&LBME=1, answering the corresponding received frames and error frames is prohibited, but data can be sent. Note: Setting this bit is prohibited in communication.
13	STBY	rw	0	Transceiver standby mode 0 - Disable 1 - Enable This register bit is connected to the output signal stby and can be used to control the standby mode of the transceiver. STBY cannot be set to 1 if TPE=1, TSONE=1, or TSALL=1. If the host sets STBY to 0, then the host needs to wait the time required for the transceiver to start before the host requests a new transmission.
12	TPE	rw	0	Transmit Primary Enable 0: Disable PTB transmission

Bit	Name	R/W	Reset Value	Function
				1: Enables PTB sending When this bit is enabled, the Mailbox in the PTB will be sent at the next available location. The STB transmission already started will continue, but the next waiting STB transmission will be delayed until the PTB transmission is completed. This bit will remain 1 after a write 1 until the PTB send is complete or until the send is cancelled via TPA. Software cannot clear this bit by writing a 0. TPE is reset to the reset value by hardware in the following cases: RESET=1 BUSOFF=1 LOM=1&LBME=0 TTEN=1&TTTBM=1 ROP=1
11	TPA	rw	0	Transmit Primary Abort 0: No cancellation 1: Cancels a PTB transmission that has been requested via TPE Set 1 but has not yet started This bit must be set by the host controller and reset by CAN-CTRL. Set TPA automatically cancels the set TPE. the host controller can set TPA to 1 but cannot reset it to 0. the host cannot set it for the short time that CAN-CTRL resets this bit. This bit will reset to a hardware reset value if RESET=1 or (TTEN=1 and TTTBM=1). tpa should not be set at the same time as tpe.
10	TSONE	rw	0	Transmit Secondary ONE frame 0: Do not send 1: Send one frame of STB data In FIFO mode, the first written data is sent, and in priority mode, the highest priority data is sent. After writing 1, this bit will remain 1 until STB transmission is completed or cancellation of transmission by TSA. Software cannot clear this bit by writing 0. TSONE is reset to the reset value by hardware in the following cases: RESET=1 BUFOFF=1 LOM=1&LBME=0 TTEN=1&TTTBM=1 ROP=1
9	TSALL	rw	0	Transmit Secondary ALL frame 0: Do not send

Bit	Name	R/W	Reset Value	Function
				1: Send all the data in STB This bit will remain 1 after writing 1 until STB transmission is completed or cancellation of transmission by TSA. Software cannot clear this bit by writing 0. TSALL is reset to the reset value by hardware in the following cases: RESET=1 BUSOFF=1 LOM=1&LBME=0 TTEN=1&TTTBM=1 If a new frame is loaded by the STB during transmission, the new frame will also be transmitted.
8	TSA	rw	0	Transmit Secondary Abort 0: No cancellation 1: Cancels STB transmissions that have been requested by TSONE or TSALL set to 1 but have not yet started This bit is written to 1 by software but cleared by hardware. Writing 1 clears the TSONE or TSALL bit. TSA is reset to the reset value by hardware in the following cases: RESET=1 BUSOFF=1 TSA should not be set at the same time as TSONE or TSALL.
7	RESET	rw	1	Reset request bit 0: Do not request a partial reset 1: Request local reset Some registers can only be written when RESET=1, please refer to the software reset function, when the node enters the BUS OFF state, the hardware will automatically RESET position 1. Please note that when RESET=0 it takes 11 CAN bit times for the node to participate in communication.
6	LBME	rw	0	External loopback mode enable bit 0: Disable external loopback mode 1: Enables external loopback mode Note: Setting this bit is prohibited in communication.
5	LBMI	rw	0	Internal loopback mode enable bit 0: Disable internal loopback mode 1: Enables internal loopback mode Note: Setting this bit is prohibited in communication.
0	BUSOFF	rw	0	Bus Off 1 - Controller status is "Bus Off". 0 - Controller status is "Bus On".

Bit	Name	R/W	Reset Value	Function
				Writing 1 to BUSOFF will reset TECNT and RECNT. This should only be used for debugging.

24.4.9. Error Warning Register (CANFD_WECR)

Address offset：0x2c

Reset value：0x0000001b

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECNT[7:0]								RECNT[7:0]							
r	r	r	r	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
KOER[2:0]			ALC[4:0]					AFWL[3:0]				EWL[3:0]			
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:24	TECNT	r	0	Transmit Error Count The transmit error counter increases or decreases according to the error count specified by the CAN protocol. The counter does not overflow and 255 is the maximum value.
23:16	RECNT	r	0	Receive Error Count The receive error counter increases or decreases according to the error count specified by the CAN protocol. The counter does not overflow and 255 is the maximum value.
15:13	KOER	rw	0	Kind Of Error 000: No error 001: Bit Error 010: Form Error 011: Padding Error 100: Response Error 101: CRC error 110: Other errors 111: Reserved The KOER bit is updated when there is an error, and the KOER bit remains unchanged when sending and receiving normally.
12:8	ALC	r	0	Arbitration Lost Capture (ALC) ALC records the position of a frame of data at the time of arbitration failure.
7:4	AFWL	rw	0x1	Receive buffer Almost Full Warning Limit The set value range is 1~3. AFWL=0 is meaningless and is treated as AFWL=1.

Bit	Name	R/W	Reset Value	Function
3:0	EWL	rw	0xb	Programmable Error Warning Limit Error Waring Limit=(EWL+1)*8. The value of this register setting affects the EIF flag.

24.4.10. Reference ID register (CANFD_REFMSG)

Address offset：0x30

Reset value：0x0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REF_IDE	Res.	Res.	REF_ID[28:16]												
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
REF_ID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31	REF_IDE	rw	0	REFerence message IDE bit 0: Standard format 1: Extended format
28:0	REF_ID	rw	0	REFerence message IDentifier REF_IDE=0: REF_ID[10:0] is valid REF_IDE=1: REF_ID[28:0] is valid REF_ID is used to detect the reference message and is applicable for both transmit and receive. When a reference message is detected, the Sync_Mark of the current frame becomes Ref_Mark. REF_ID[2:0] is fixed to 0 and its value is not checked, so that up to 8 potential time masters can be supported. When the highest byte of REF_MSG is written, it is necessary to wait for 6 CAN clock cycles to complete the transfer of REF_MSG to the CAN clock domain.

24.4.11. TTCAN Configuration Register (CANFD_TTCR)

Address offset：0x34

Reset value：0x0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	T_PRESC[1:0]		TTEN	TBE	TBF	TBPTR[5:0]					

					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
TEW[3:0]				Res.	TTYPE[2:0]			Res.	Res.	TTPTR[5:0]					
rw	rw	rw	rw		rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
26:25	T_PRESC	rw	0	TTCAN Timer PRESScaler 00b - 1 01b - 2 10b - 4 11b - 8 The TTCAN time base is the CAN bit time defined by PRESC, AC_SEG_1 and AC_SEG_2. The use of T_PRESC defines an additional prescaling factor of 1, 2, 4 or 8. T_PRESC can only be modified when TTEN=0, but it is possible to modify T_PRESC and set TTEN with one write access at the same time.
24	TTEN	rw	0	Time Trigger Enable 0: Disable 1: Enable TTCAN, the counter starts counting.
23	TBE	rw	0	Set TB slot to "empty" 0: No operation 1: The SLOT selected by TBPTR is marked as empty When SLOT is marked as empty and TSFF=0, TBE is automatically reset to 0. If this bit is set to 1, TBE=1 if there is data being sent in the selected SLOT, then TBE resets to 0 when sending is complete, sending is wrong or sending is canceled. TBE has higher priority than TBF.
22	TBF	rw	0	Set TB slot to "Filled" 0: No operation 1: The SLOT selected by TBPTR is marked as filled When SLOT is marked as filled and TSFF=1, TBE is automatically reset to 0.
21:16	TBPTR	rw	0x00	Pointer to a TB message slot 0x00 - Pointer to a PTB Other - pointer to a slot in the STB The frame slot pointed to by TBPTR can be read/written using the TBUF register.

Bit	Name	R/W	Reset Value	Function
				Write access is only possible when TSFF=0. Setting TBF to 1 marks the selected slot as filled and setting TBE to 1 marks the selected slot as empty. TBSEL and TSNEXT are not used in TTCAN mode and have no meaning. TBPTR can only point to buffer slots that exist in hardware. The unavailable bit of TBPTR is fixed to 0. TBPTR is limited to PTB and 63 STB slots. No more slots can be used in TTCAN mode. If TBPTR is too large and points to an unavailable slot, TBF and TBE will be automatically reset and no operation will be performed.
15:12	TEW	rw	0	Transmit Enable Window Single Shot Transmit Trigger mode for TTCAN, you can set TEW+1 cycle time window, transmitting is only allowed in this window.
10:8	TTYPE	rw	0	Trigger Type 000: Immediate Trigger for immediate transmission 001: Time Trigger for receive triggers 010: Single Shot Transmit Trigger for exclusive time windows 011: Transmit Start Trigger for merged arbitrating time windows 100: Transmit Stop Trigger for merged arbitrating time windows Others: Reserved The trigger time is set by TT_TRIG register and TB Slot is selected by TTPTR.
5:0	TTPTR	rw	00	Transmit Trigger TB slot Pointer 000: Pointing to PTB 001: pointing to STB SLOT1 010: point to STB SLOT2 011: Pointing to STB SLOT3 Others: set forbidden If the pointing TB SLOT is marked as empty, TEIF is set when the trigger time is reached.

24.4.12. TTCAN Trigger Register (CANFD_TTTR)

Address offset：0xffff0000

Reset value：0x02020003

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

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:16	TT_WTRIG	rw	0xffff	Watch trigger time TT_WTRIG(15:0) Defines the cycle time of the watch trigger. The initial watch trigger is the maximum cycle time 0xffff.
15:0	TT_TRIG	rw	0	Trigger time TT_TRIG(15:0) defines the cycle time of the trigger. For transmission triggering, the earliest transmission point of the SOF of the corresponding frame will be TT_TRIG+1.

24.4.13. CANFD_SCMS

Address offset：0x3c

Reset value：0x0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HELOC[2:0]		TXB	TXS	ACFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			r	r	r	r	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSTIM[2:0]			Res.
												rw		rw	

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	-
28:27	HELOC	r	0	Host-side memory error location 00 - No error when accessing from host side 01 - Error accessing from host side in TBUF 10 - Error accessing from host side in RBUF 11 - Error on access from host side in ACF During the read access from the host side, HELOC will be updated with each new error. This is sufficient since read errors during read access from the CAN side will be signaled by ACFA, TXS and TXB. The HELOC will only be updated in case of errors, but not in case of warnings caused by corrected single-bit errors.
26	TXB	r	0x0	Transmission Block 0 - Normal operation 1 - Transmission blocked

Bit	Name	R/W	Reset Value	Function
				If MDEIF or MAEIF is set due to an error while the CAN protocol machine is reading data for transmission, the transmission will be blocked immediately. If SEIF is set, the transmission is also blocked immediately. If TXB=1, the error counter is frozen. If reset=1, TXB is reset.
25	TXS	r	0	Transmission stop 0 - Normal operation 1 - Transfer stopped If MDEIF or MAEIF is set due to an error during a priority reorder computer access to memory, any new transfers will stop. If there is an active transfer, this will complete before stopping, but if an error occurs during this transfer, no re-transmission will begin. If reset = 1, TXS is reset.
24	ACFA	rw	0	Accept filter reception 0 - normal operation of ACF 1 - ACF disable: all received frames are accepted If MDEIF or MAEIF is set due to an error in the ACF address range, ACFA is set. then the Accept filter is disabled and all frames will be accepted. ACFA can be reset by writing a 1 similar to the interrupt flag. However, since ACFA will be set while the receive is still active, it needs to be reset after the receive is complete and the RIF is set. If reset = 1, ACFA will also be reset.
3:1	FSTIM	rw	0x000	Fault stimulation Fault stimulation is only possible when XMREN=1.
0	Res	-	-	-

24.4.14. Filter Group Control Register (CANFD_ACFCR)

Address offset：0x44

Reset value：0x00010000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AE_11	AE_10	AE_9	AE_8	AE_7	AE_6	AE_5	AE_4	AE_3	AE_2	AE_1	AE_0
				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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACFADR			
												rw	rw	rw	rw

Bit	Name	R/W	Reset Value	Function
31:16	AE_x	rw	0x1	ACF Enable 1 - Enable 0 - Disable Each acceptance filter (ACFC / ACFM) can be enabled or disabled individually. After a hardware reset, only filter number 0 is enabled by default. A disabled filter rejects a frame. If the appropriate ACFC / ACFM configuration matches, only the enabled filter will accept frames.
3:0	ACFADR	rw	0	Filter Address ACFADR points to a specific acceptance filter. The selected filter can be accessed using registers ACFC and ACFM. The value ACFADR>ACF_NUMBER-1 is nonsensical and is automatically treated as the value ACF_NUMBER-1

24.4.15. Filter group code register (CANFD_ACFC)

Address offset：0x48

Reset value：0xXXXXXXXX

0x00	CANFD_ID
0x04	CANFD_FORMAT
0x08	CANFD_TYPE

Note: Please refer to the definition of LLC frame format for the control meaning of the above registers

24.4.16. Filter groupmask register (CANFD_ACFM)

Address offset：0x58

Reset value：0xXXXXXXXX

0x00	CANFD_ID
0x04	CANFD_FORMAT
0x08	CANFD_TYPE

Note: Please refer to the definition of LLC frame format for the control meaning of the above registers

24.4.17. CAN Receive BUF register (CANFD_RBUF)

Address offset：0x70

Reset value：0xXXXXXXXX

Note: Please refer to the definition of LLC frame format for the control meaning of the above registers

24.4.18. CAN transmit BUF register (CANFD_TBUF)

Address offset：0xcc

Reset value：0xXXXXXXXX

Note: Please refer to the definition of LLC frame format for the control meaning of the above registers

24.4.19. CAN register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CANFD_ID	Res.	Res.	Res.	ID[28:0]																												
0x04	CANFD_FORMAT	Res.	Res.	Res.	LBF	ESI	KOER[2:0]			Res.	Res.		RMF		BRS	FDF	IDE	Res.	Res.	Res.	Res.	Res.	DLC[10:0]										
0x08	CANFD_TYPE	HANDLE[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																
0x0C	CANFD_AF	AF[31:0]																															
0x18	CANFD_TTCAN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CYCLE_TIME[15:0]															
0x1C	CANFD_DATA1	data[31:0]																															
.....	.....	.....																															
0x58	CANFD_DATA16	data[31:0]																															

25. Serial Peripheral Interface (SPI)

25.1. Introduction

The SPI interface can be configured to support the SPI protocol or to support the I2S audio protocol. The SPI interface operates in SPI mode by default, and the functionality can be switched from SPI to I2S mode by software.

The Serial Peripheral Interface (SPI) allows the chip to communicate with external devices in half/full duplex, synchronous, serial mode. This interface can be configured in master mode and provides a communication clock (SCK) for external slave devices. The interface can also operate in a multi-master configuration. It can be used for a variety of purposes, including two-wire simplex synchronous transmission using one bi-directional data line, and also reliable communication using CRC checksum.

I2S is also a 3-pin synchronous serial interface communication protocol. It supports four audio standards, including the Philips I2S standard, the MSB and LSB alignment standards, and the PCM standard. It can operate in 2 modes, master and slave, in half-duplex communication. When it acts as a master device, it provides a clock signal to the external slave device through the interface.

25.1.1. Main features

25.1.1.1. SPI Features

- 3-Wire Full Duplex Synchronous Transmission
- Duplex simplex simultaneous transmission with or without a third bi-directional data line
- 8 or 16-bit transmission frame format selection
- Master or slave operation
- Supports multi-master mode
- 8 master mode baud rate prescale factors (up to fPCLK/4)
- Slave mode frequency (up to fPCLK/4)
- Fast communication between master and slave modes
- NSS management by software or hardware in both master and slave modes: dynamic change of master/slave operation mode
- Programmable clock polarity and phase
- Programmable data order, MSB first or LSB first
- Dedicated transmit and receive flags that can trigger interrupts
  - SPI bus busy status flag
- Hardware CRC to support reliable communication
  - In transmit mode, the CRC value can be sent as the last byte
  - CRC checksum is automatically performed on the last byte received in full duplex mode
- Interrupt-triggered master mode faults, overloads, and CRC error flags
- Two 64 (16x4) bit embedded Rx and Tx FIFOs with DMA functionality
- DMA-capable 1-byte transmit and receive buffers: generates transmit and receive requests

25.1.1.2. I2S function

- Simplex communication (send or receive only)
- Master or slave operation
- 8-bit linear programmable prescaler for accurate audio sampling frequencies (8KHz to 96kHz)
- Data format can be 16-bit, 24-bit or 32-bit
- Audio channel fixed packet frames of 16 bits (16-bit data frames) or 32 bits (16, 24 or 32-bit data frames)
- Programmable clock polarity (steady state)
- Underflow flag bit in slave transmit mode and overflow flag bit in master/slave receive mode
- 16-bit data registers for transmit and receive, one at each end of the channel
- Supported I2S protocols:
  - I2S Philips standard
  - MSB alignment standard (left-aligned)
  - LSB alignment standard (right-aligned)
  - PCM standard (16-bit channel frame with long or short frame synchronization or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for both transmit and receive
- Master clock can be output to external audio devices with a fixed ratio of 256xFs (Fs is the audio sampling frequency)

25.1.2. Module Block Diagram

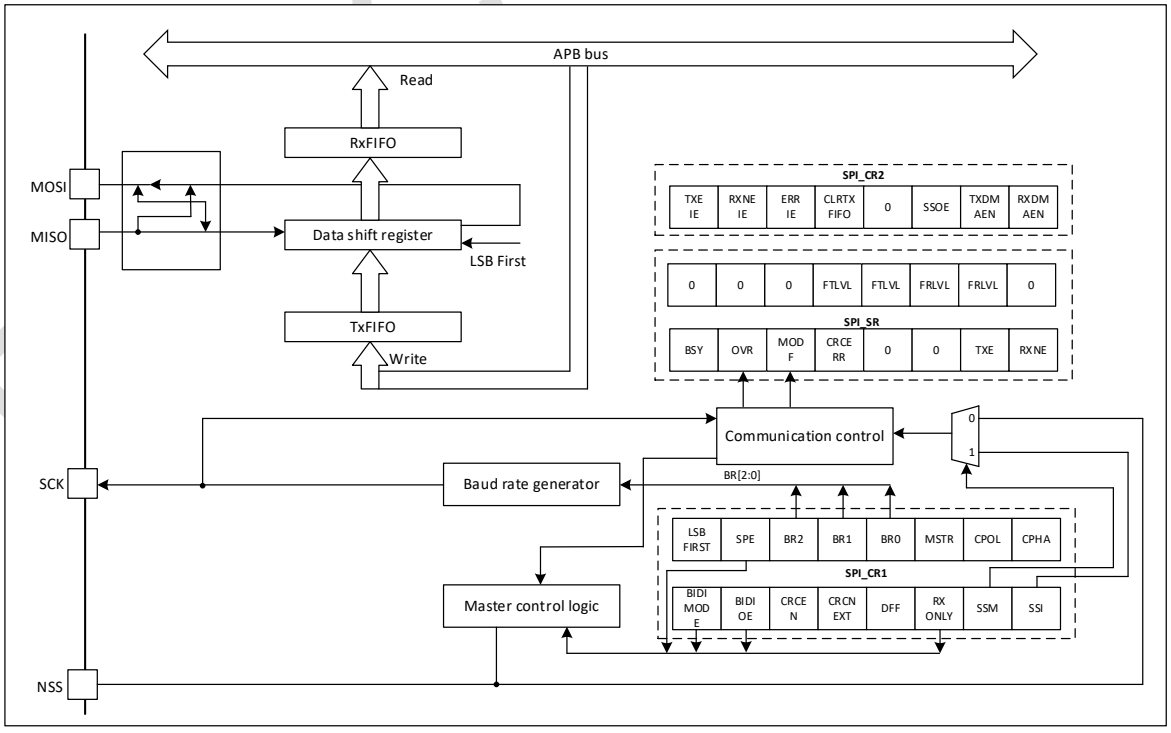


Figure 25-1 SPI Module

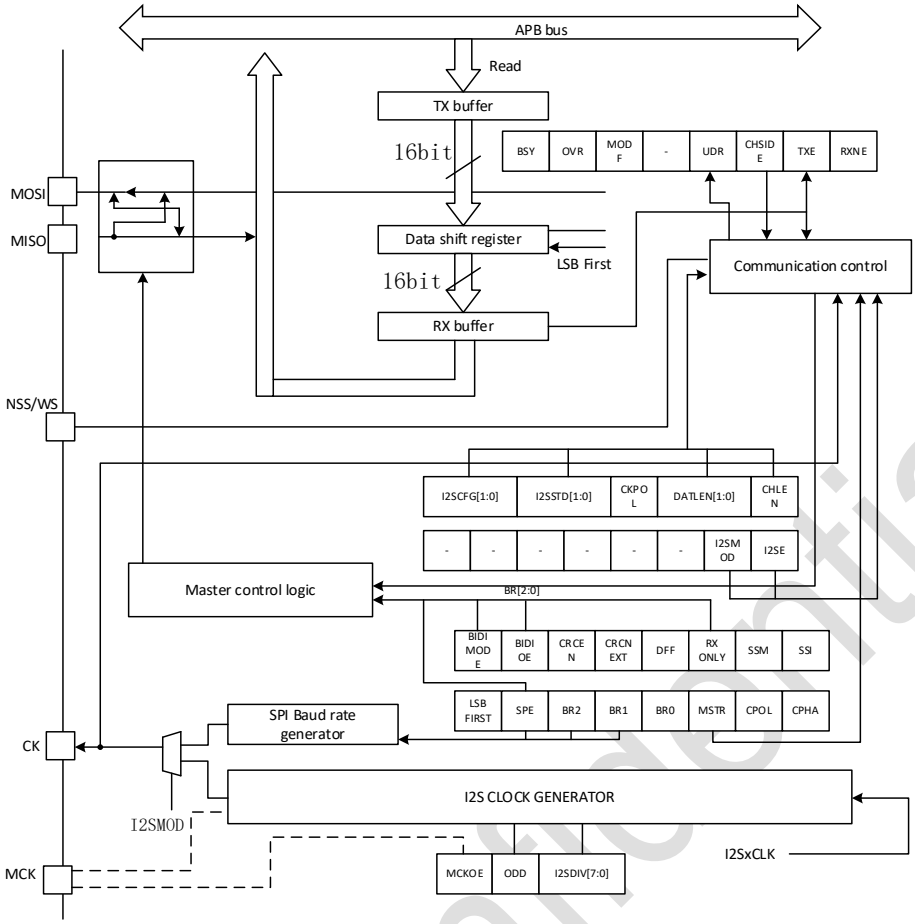


Figure 25-2 I2S Module

25.2. SPI Function Description

25.2.1. Overview

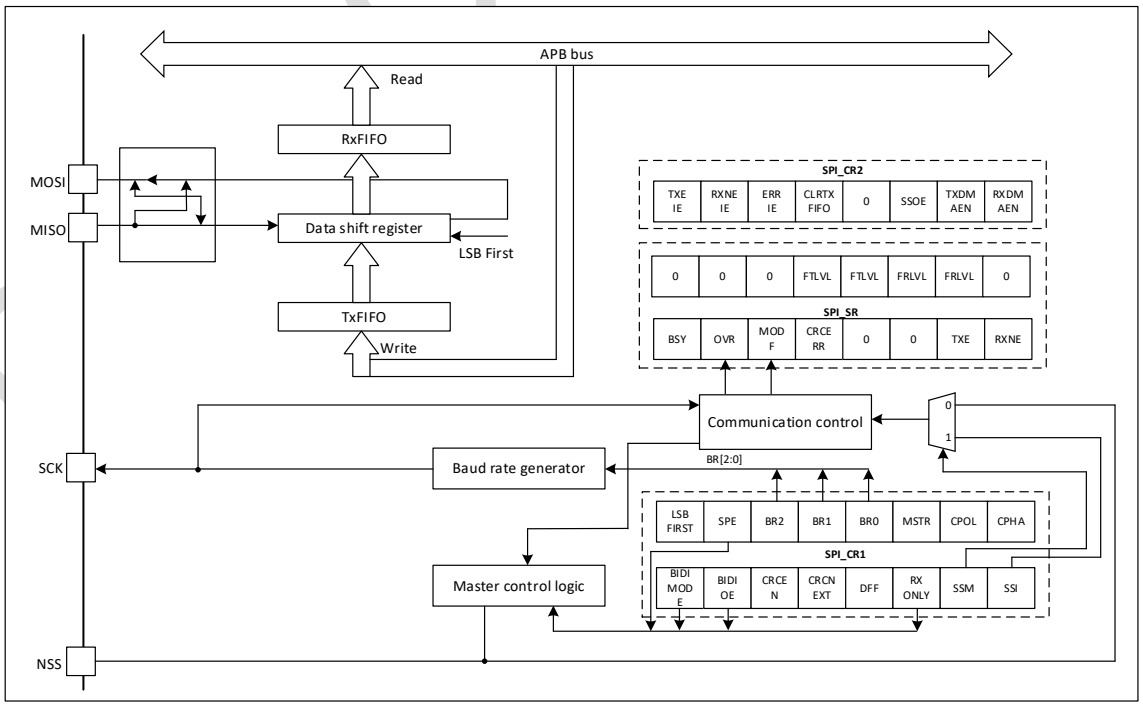


Figure 25-3 SPI Module

The SPI is connected to external devices through four pins:

MISO: Master device input/slave device output pin. This pin sends data in slave mode and receives data in master mode.

MOSI: Master device output/slave device input pin. This pin sends data in master mode and receives data in slave mode.

SCK: Serial clock, which is used as the output of the master device and the input of the slave device.

NSS: Slave device selection. Depending on the SPI and NSS settings, this pin can be used as:

- Select the slave to communicate with
- Synchronize data frames
- Conflict found between multiple masters

The SPI bus allows communication between a master and one or more slaves. The bus consists of at least two lines: one for the clock and the other for the data to be transmitted synchronously. Depending on the application scenario, an additional data line and slave select signal can be optionally added.

25.2.2. Single-master and single-slave communication

For different application scenarios, the SPI can communicate using several different configurations. These configurations use 2-wire, 3-wire (software NSS management), or 3-wire, 4-wire (hardware NSS management). The communication is usually initiated by the host.

25.2.2.1. Full Duplex Communication

By default, SPI is configured for full-duplex communication. In this configuration, the shift registers of the host and slave, between MOSI and MISO, are connected together using two unidirectional lines. During SPI communication, data is shifted synchronously along the clock provided by the host. The host sends data through MOSI and receives data from the slave through MISO. When the data frame transfer is complete (all bits are shifted), the information between the host and the slave has been interacted with.

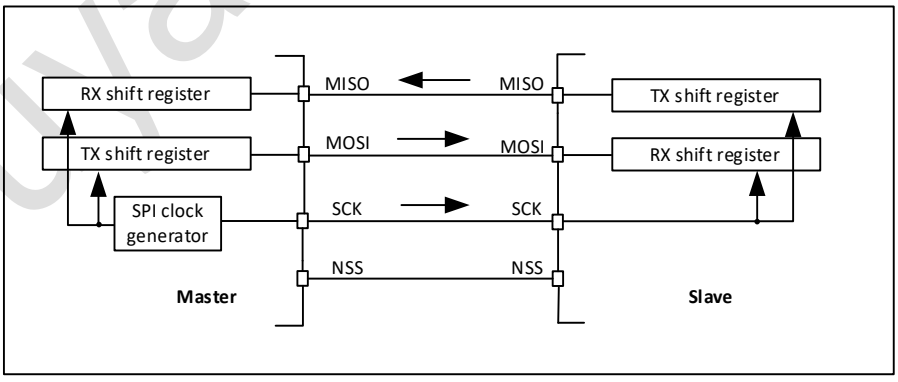


Figure 25-4 Full duplex single host/single slave applications

25.2.2.2. Half-duplex communication

The SPI can operate in half-duplex mode by setting the BIDIMODE bit (SPI_CR1 register). In this configuration, the connection between the host and slave shift registers is done with 1 data line. During communication, the data is shifted synchronously between the two shift registers in the

direction selected by BIDIOE (SPI_CR1 register) at the clock edge of SCK. In this configuration, the MISO of the master and MOSI of the slave are released as general-purpose ports for other applications.

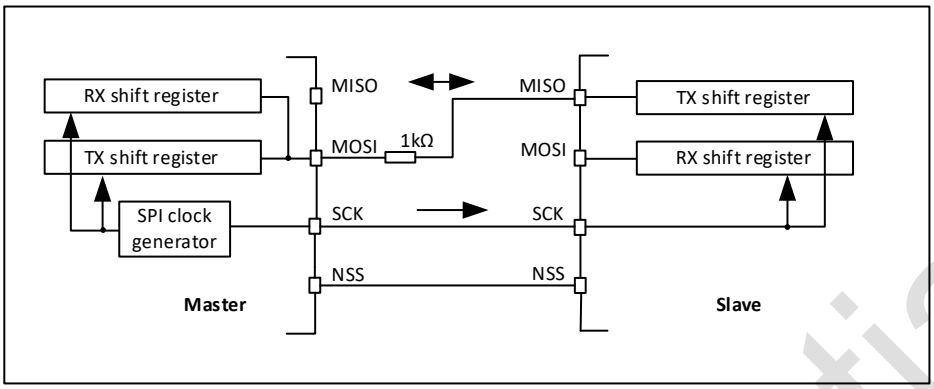


Figure 25-5 Half-duplex single host/single slave applications

NSS can be used to perform hardware control flow between the host and slave. Optionally, NSS can also be used without it. The flow is then to be processed internally. In this configuration, the MISO of the host and the MOSI of the slave can be used as generic ports for other applications.

25.2.2.3. Simplex communication

Set the SPI in transmit-only mode or receive-only mode by using RXONLY (SPI_CR1 register) to make the SPI operate in single-wire mode. In this configuration, only 1 wire is used between the shift registers of the host and the slave. The other pair, MISO and MOSI, is not used and can be released as a general purpose port.

- Send-only mode (RXONLY=0): Configuration is the same as full duplex. The application ignores messages on the unused port. This port can be used as a standard GPIO.
- Receive-only mode (RXONLY=1): By setting RXONLY, the application can disable the SPI output function. In the slave configuration, the MISO output is disabled and the port is used as a GPIO. the slave continues to receive data from the MOSI when his slave NSS signal is active. Whether the received data event occurs or not depends on the configuration of the data buffer. In the host configuration, the MOSI output is disabled and the port can be used as a GPIO. the clock signal is generated continuously as long as the SPI is used. The only way to stop the clock is to clear RXONLY or SPE until the input from the MISO is complete.

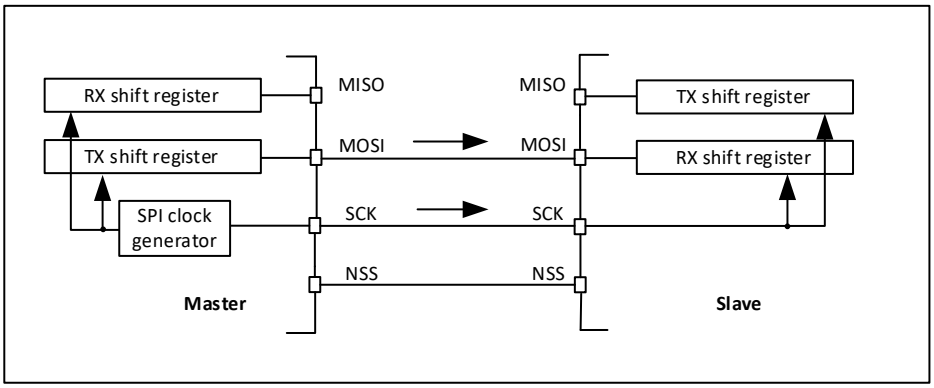


Figure 25-6 Simplex single slave/single host applications

(Master device in transmit-only mode / Slave mode in receive-only mode)

1) NSS can be used for hardware control flow between the host and the slave. Optionally, NSS can also be used without it. The flow is then to be processed internally.
2) The input in the Rx shift register captures unexpected input information. In the standard transmit-only mode, all events related to the transmission reception are must be ignored.
3) In this configuration, both MISO pins are used as GPIOs.

By setting with the transmission direction (bidirectional mode is enabled when the BIDIO bit is not changed), any simplex communication can be replaced by half-duplex communication.

25.2.3. Multi-slave communication

In a configuration with two or more independent slaves, the host manages the NSS for each slave, using GPIOs. the host must select a particular slave by pulling down the connected slave NSS. When this is done, standard host and dedicated slave communication is established.

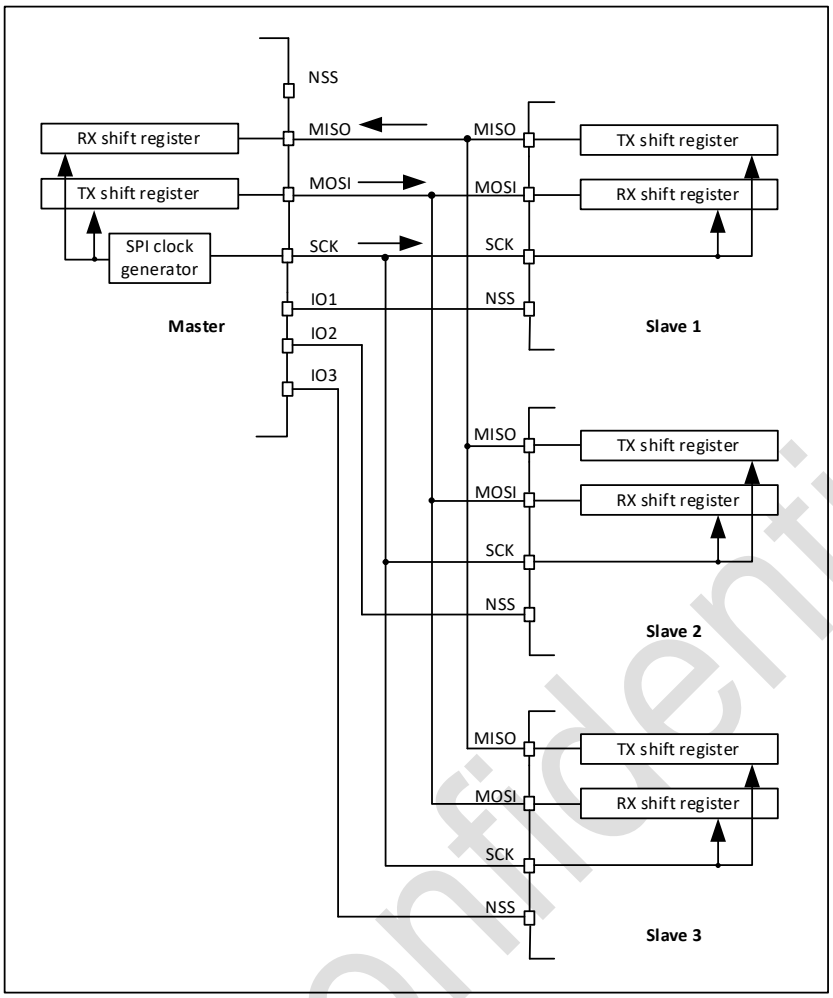


Figure 25-7 Host communicates with three independent slaves

NSS is not used on the host side in this configuration. Any MODF errors must be prevented by SSM=1, SSI=1.

Since the slave MISOs are connected together, all slaves must configure the GPIOs of their MISOs as AF open-drain.

25.2.4. Multi-host communication

Unless the SPI bus is not designed to be multi-host capable, the user can use its embedded FEATURE, which detects a potential conflict between two nodes trying to control the bus at the same time. When this detection is required, use the NSS pin configured for hardware input mode.

Connections with more than two SPI nodes working in this mode are not possible, because only one node can put outputs on the common data line at the same time.

When the node is invalid, by default both remain in slave mode. Once the node wants to control the bus, it switches itself into master mode and gives the valid level to the slave select input of the remaining node via a dedicated GPIO. after the process is finished, the valid select signal is released and the node controlling the bus temporarily returns to passive mode and waits for a new process to start.

If both nodes give control requests at the same time, a bus conflict event is generated (check MODF events). The user can then apply some simple arbitration process (e.g., defer the next attempt by giving different timeouts to the two nodes defined in advance)

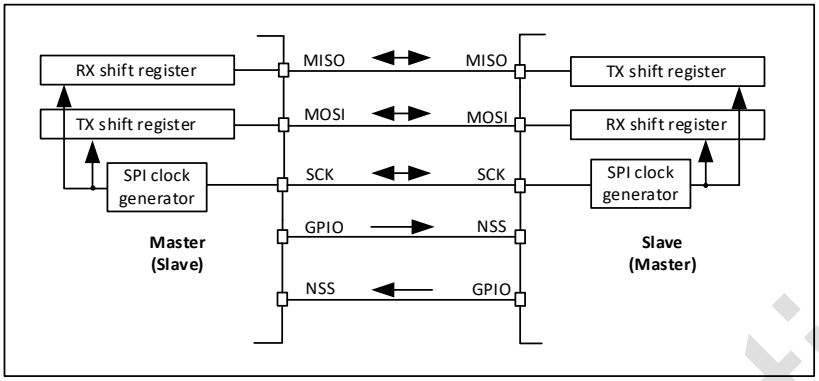


Figure 25-8 Multi-host applications

The NSS pin is configured in hardware input mode at both nodes. His active level enables the MISO output control, while the passive node is configured as a slave.

25.2.5. From Selection (NSS) Pin Management

In slave mode, the NSS is used as a standard chip select input to enable the slave to communicate with the host. In master mode, NSS can be used as both an output and an input. When used as an input, it prevents bus conflicts for multiple masters, and when used as an output, it drives the slave select signal for a single slave.

The SSM bit of the SPI_CR1 register allows the selection of hardware or software slave management:

- Software NSS Management (SSM=1): In this configuration, the slave select signal is driven by the internal SSI bit (SPI_CR1 register) value. The external NSS pin is released for use by other applications.
- Hardware NSS management (SSM=0): In this case, there are several possible configurations.

  1) NSS output enable (SSM=0, SSOE=1): This configuration is used only when acting as the host. The hardware manages the NSS pin. when the SPI is enabled in host mode (SPE=1), the NSS signal is pulled low and held low until the SPI is disable (SPE=0). In multi-host applications, the SPI cannot be configured for this NSS configuration.

  2) NSS output disable (SSM=0, SSOE=0): This configuration allows for multi-host capability if the MCU is acting as the host on the bus. If the NSS pin is pulled low at this point, the SPI enters the host mode fault state and the chip is automatically reconfigured to slave mode. In slave mode, the NSS pin is used as a standard chip select input and the slave is selected when the NSS is low.

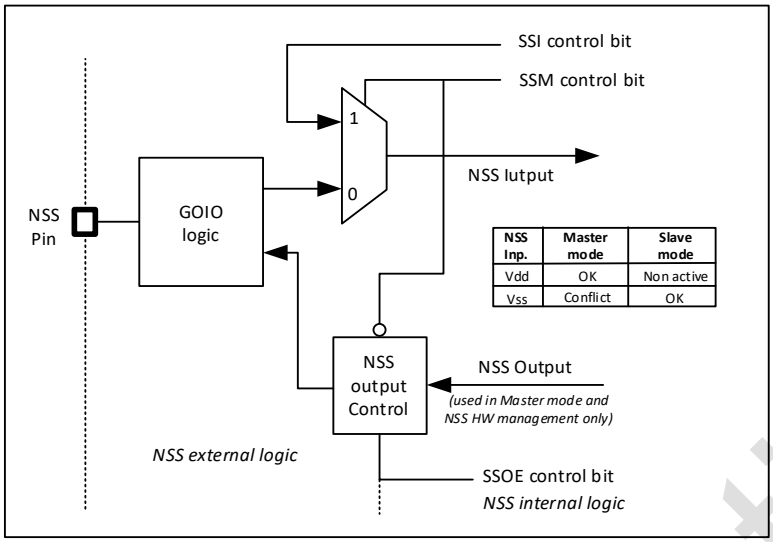


Figure 25-9 Hardware/software slave selection management

25.2.6. Communication Format

During SPI communication, the receive and transmit operations are performed simultaneously, and the SCK (serial clock) synchronizes the information shifting and sampling operations on the data lines. The communication format depends on the clock phase, clock polarity, and data frame format. In order to be able to communicate, the host and slave must follow the same communication format.

25.2.6.1. Clock phase and clock polarity control

With the CPOL and CPHA bits (SPI_CR1 register), the software can configure four possible timings. CPOL (clock polarity) controls the IDLE state of the clock when no data is being transferred. This bit affects both the host and the slave. If CPOL is reset, the SCK pin has a low state. If CPOL is set, the SCK pin has a high IDLE state.

If CPHA is set, the second edge of SCK captures the first data bit transmitted (falling edge if CPOL is reset, rising edge otherwise). At the appearance of the clock change type, the data is latched. If CPHA is reset, the first edge of SCK captures the first transmitted data bit (a falling edge if CPOL is set, otherwise a rising edge). At the appearance of this clock change type, the data is latched.

The combination of CPOL and CPHA selects the data capture clock edge.

The SPI must be disable (SPE=0) before CPOL/CPHA can be changed.

The IDLE state of SCK must correspond to the polarity selected by the SPI_CR1 register.

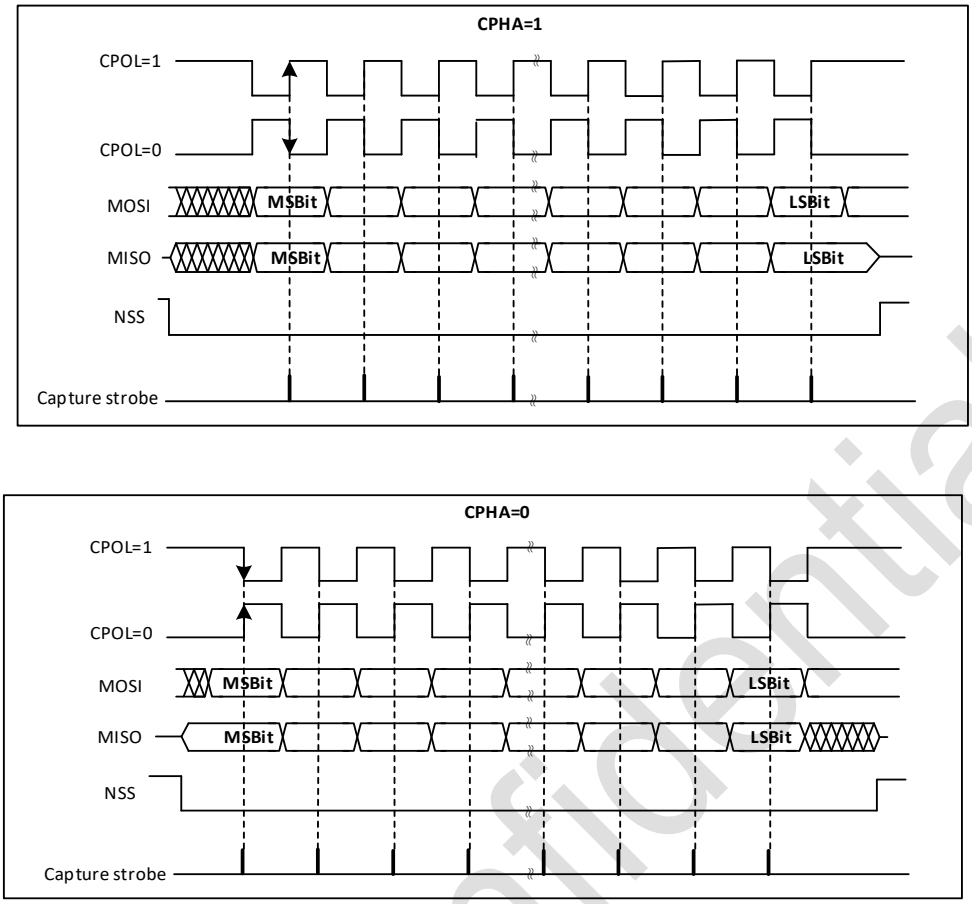


Figure 25-10 Data Clock Timing Diagram

The order of the data bits depends on the setting of the LSBFIRST bit.

25.2.6.2. Data frame format

The SPI shift register can be set to MSB-FIRST or LSB-FIRST by using the LSBFIRST bit (SPI_CR1 register). the number of bits of the data frame is selected by using the DFF bit (SPI_CR1 register). It can be selected as 8-bit or 16-bit length, and this setting is applicable for both transmit and receive.

25.2.7. SPI Configuration

For both hosts and slaves, the SPI configuration process is almost identical. For specific mode establishment, follow the special section on this. When performing standard communication, perform the following steps:

- Write related GPIO registers: configure MOSI, MISO and SCK pins
- Write SPI_CR1 register
  - Configure clock baud rate via BR[2:0] (not required for slave mode)
  - Configure CPOL and CPHA
  - Select simplex or half-duplex mode via RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE cannot be active at the same time)
  - Configure LSBFIRST
  - Configure SSM and SSI

- Configure MSTR bit (in a multihost NSS configuration, if the host is configured to prevent MODF errors, to avoid conflicting NSS states)
- Configure the DFF bit and select the number of data frame bits

- Write SPI_CR2 register
  - Configure SSOE (not required for slave mode)
- Write the corresponding DMA registers: Configure the SPI Tx and Rx channels for DMA

25.2.8. SPI Enable Flow

It is recommended that the SPI slave be enabled before the host sends the clock. If this is not done, undesired data transfers may occur. The slave's data register must already contain the data to be sent before it starts communicating with the host (either at the first edge of the communication clock, or if the clock signal is continuous, before the end of ongoing communication). the SCK signal must be fixed at IDLE state level (with the corresponding selected polarity) before the SPI slave is enabled.

In full duplex mode (or transmit-only mode), the host starts communication when the SPI is enabled and the TXFIFO is not empty, or the next write to the TXFIFO is made.

In any host receive-only mode (RXONLY=1, or BIDIMODE=1 and BIDIOE=0), after the SPI is enabled, the master starts communication and the clock is immediately provided.

For DMA processing, follow the specific section.

25.2.9. Data transmission and reception process

25.2.9.1. Receive and send buffers

All SPI data communication is through FIFOs. This feature allows the SPI to work with a continuous data stream and prevents communication problems caused by the CPU not having time to process the data. There are separate FIFOs for transmit and receive, called TXFIFO and RXFIFO. these FIFOs are used in all SPI modes.

FIFO processing depends on a variety of parameters including: data exchange mode (full duplex, half duplex), data frame format, and the size of the access FIFO data registers (8-bit or 16-bit).

Reading the SPI_DR register will result in the earliest data stored in the RXFIFO that has not yet been read away. Writing the SPI_DR register will deposit the data being written at the end of the FIFO transmit queue. Read accesses must normally be aligned with the RXFIFO threshold, and the FTLVL[1:0] and FRLVL[1:0] bits show the current occupancy levels of the two FIFOs.

All accesses to the SPI_DR register must be managed through the RXNE event. This event is triggered when the data memory is non-empty at the RXFIFO. When RXNE is cleared, the RXFIFO is considered empty.

Similarly, writing the data frames to be sent is managed through the TXE event. This event is triggered when the TXFIFO Level is less than or equal to half of the total capacity. Otherwise, TXE is cleared and the TXFIFO is considered full.

With this approach, both RXFIFO and TXFIFO can store 4 data frames, while TXFIFO can store only 3 data frames (when the data frame format is not larger than 8bit). This difference prevents the

following confusing situation: three 8-bit data frames already exist in the TXFIFO, while the software has to write data to the TXFIFO in 16-bit (CPU data width) mode again.

Both TXE and RXNE events can be handled by queries, interrupts and DMA.

The overrun event is generated when the RXFIFO is full, if the next data is received. overrun events can be handled by query and interrupt.

The BSY bit that is set indicates that communication is in progress for 1 current data frame. When the clock signal is provided continuously, the BSY flag remains set between two data frames on the master side. However, between each data frame transmission on the slave side, the BSY remains low for a minimum of 1 SPI Clock width.

In some application scenarios, when data is written to the TXFIFO, the TXFIFO data can be cleared by setting the CLRTXFIFO bit, so that new data can be written to the TXFIFO to resume communication.

25.2.9.2. Sequential processing

Some data frames can be passed in a single sequence to complete a message. When transmit is enabled and when the host has any data in the TXFIFO, the sequence starts and continues. The clock signal is continuously supplied by the host until the TXFIFO is empty, then it stops waiting for additional data.

In receive-only mode, i.e. half-duplex (BIDIMODE=1, BIDIOE=0) or simplex mode (BIDIMODE=0, RXONLY=1), the host starts transmitting immediately after the SPI is enabled and receive-only mode is activated. The host will keep the clock available until the host stops the SPI or receive-only mode. The host receives data continuously.

When the host is able to provide all communications in a continuous mode (SCK signals are continuous), the host must take into account the slave's ability to handle the data stream. When necessary, the host must slow down the communication and provide either a slower clock, or separate frames, or reorganize packets with delays. It is important to note that for either the host or the slave, there is no underflow error signal and data from the slave is usually interacted and processed by the host (even if the slave cannot get the data ready in time). For slaves, a better approach is to use DMA, especially when the data frames are small and the baud rate is high.

Each sequence must be wrapped with NSS pulses, while one of the slaves is selected to communicate in a multi-slave system clock. In a single slave system, it is not necessary to use NSS to control the slave, but again the pulse is provided

When BSY is set, it shows the data frame interaction in progress. When the dedicated frame interaction is completed, the RXNE flag is set. The last bit is sampled and the entire data frame is stored in the RXFIFO.

25.2.9.3. Disable SPI process

When the SPI is disable, a specific disable flow must be followed. For the system disable SPI flow is important, because thereafter on the application, the peripheral clock is stopped and the system enters low power mode. In this case (disable), the ongoing interaction will be broken. In some modes, the disable process is the only way to stop continuous communication.

In full duplex or transmit-only mode, the host can complete the interaction when it stops providing the data to be sent. In this case, the clock is stopped after the last data interaction. Pay extra attention to packing mode (when interacting with an odd number of data frames to place some dummy bytes into the interaction). In these modes, the SPI is disable before the user must use the standard disable process. When the SPI is disable in host transmit, the SPI functionality is not guaranteed if a frame interaction is in progress or if the next data frame is present in the TXFIFO.

When the host is in any receive-only mode, the only way to stop the continuous clock is to stop the peripheral (SPE=0). A dedicated SPI disable process is performed in this mode.

When the SPI is disable, data received that is not read away is stored in the RXFIFO, and this data must be disposed of before the next SPI enable is to start a new sequence. To prevent unread data, make sure that the RXFIFO is empty when the SPI is disable (either by using the correct disable flow, or by initializing all SPI registers with a software reset).

The standard disable process is based on the BSY status and checking FTLVL[1:0] to ensure that the transfer is complete. It is also possible to identify the end of an ongoing interaction with a specific check, e.g:

- When the NSS signal is managed by software, the host has to provide the correct NSS pulse to the slave. Or
- When completing an interactive data stream from DMA or FIFO, when the last data frame or CRC frame is still in transit.

  The correct disable process is (except for receive-only mode)

  - Wait for FTLVL[1:0]=00 (no data to send)
  - Wait for BSY=0 (last data to be processed)
  - Disable SPI (SPE=0)
  - Read data until FRLVL[1:0]=00 (read all received data)

  For a specific receive-only pattern, the correct disable process is

  - Interrupt the receive flow by disable SPI (SPE=0) during the transmission of the last data frame
  - Wait for BSY=0 (the last data frame has been processed)
  - Read data until FRLVL[1:0]=00 (read all received data)

25.2.9.4. Using DMA communication

To work at maximum speed and to speed up the data read/write process to avoid overrun, SPI has the DMA capability of the simple request/response protocol.

When TXE or RXNE is set, a DMA request is generated. tx and Rx buffer have independent requests.

- When sending, each time TXE is set to 1, a DMA request is generated. The DMA then writes data to the SPI_DR register.
- When receiving, each time RXNE is set to 1, a DMA request is generated. Then the DMA reads the data from the SPI_DR register.

When the SPI is being used to send data only, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag bit is set because the received data is not read away. When the SPI is used only for receiving data, the SPI Rx DMA channel can be enabled.

On transmit, when the DMA has written all the data to be sent (the TCIF flag bit of the DMA_ISR register is set), the SPI communication can be ensured to be completed by monitoring the BSY flag. This is used to avoid that the final transfer is corrupted when disable SPI or enter stop mode. The software must wait for FTLVL[1:0]=00 and then wait for BSY=0.

When DMA communication is started and a DMA channel management error event occurs, the following steps must be followed:

- Enable DMA Rx buffer (RXDMAEN bit of SPI_CR2) (if Rx DMA is used)
- Enable Tx Rx DMA streams (in DMA registers) (if steams are used)
- Enable DMA Tx buffer (in TXDMAEN bit of SPI_CR2 register) (if Tx DMA is used)
- Enable SPI via SPE bit.

Force the communication to be closed with the following steps:

- Disable DMA Tx Rx streams (in DMA register) (if stream is used)
- Disable SPI via the SPI disable flow
- Disable DMA Tx and Rx buffer (if DMA Tx and/or Rx is used) by clearing TXDMAEN and RXDMAEN (SPI_CR2 registers)

25.2.9.5. Communication Timing

This section describes some typical timings that are valid for queries, interrupts or DMA. For simplicity, it is assumed that LSBFIRST=0, CPOL=0, CPHA=1. The full configuration of DMA operation is also not provided.

- When NSS is active, SPI is enabled and the slave starts to control MISO, when NSS is released or SPI is turned off the slave loses control of MISO. For the slave, sufficient time must be provided to the master before the transfer starts to prepare the data in advance.
  - On the host side, the SPI peripheral controls the MOSI and SCK signals (also the NSS signal) only when the SPI is enabled. If SPI is disable, the SPI peripheral is disconnected from GPIO, so the level values on these lines depend on the GPIO settings.
- On the host side, BSY remains active between frames if the communication is continuous. On the slave side, the BSY signal usually becomes low for at least one clock cycle between data frames.
- The TXE signal is cleared only when the TXFIFO is full.
- As soon as the TXDMAEN bit is set, the DMA arbitration process starts. After TXEIE is set, TXE interrupt is generated. When the TXE signal is valid, data transfer to the TxFIFO starts until the TxFIFO becomes full, or the DMA transfer is completed.
- If all data to be sent is loaded into the TxFIFO, the DMA Tx TCIF flag going high occurs before even SPI communication. This flag remains high until the SPI interaction is complete.

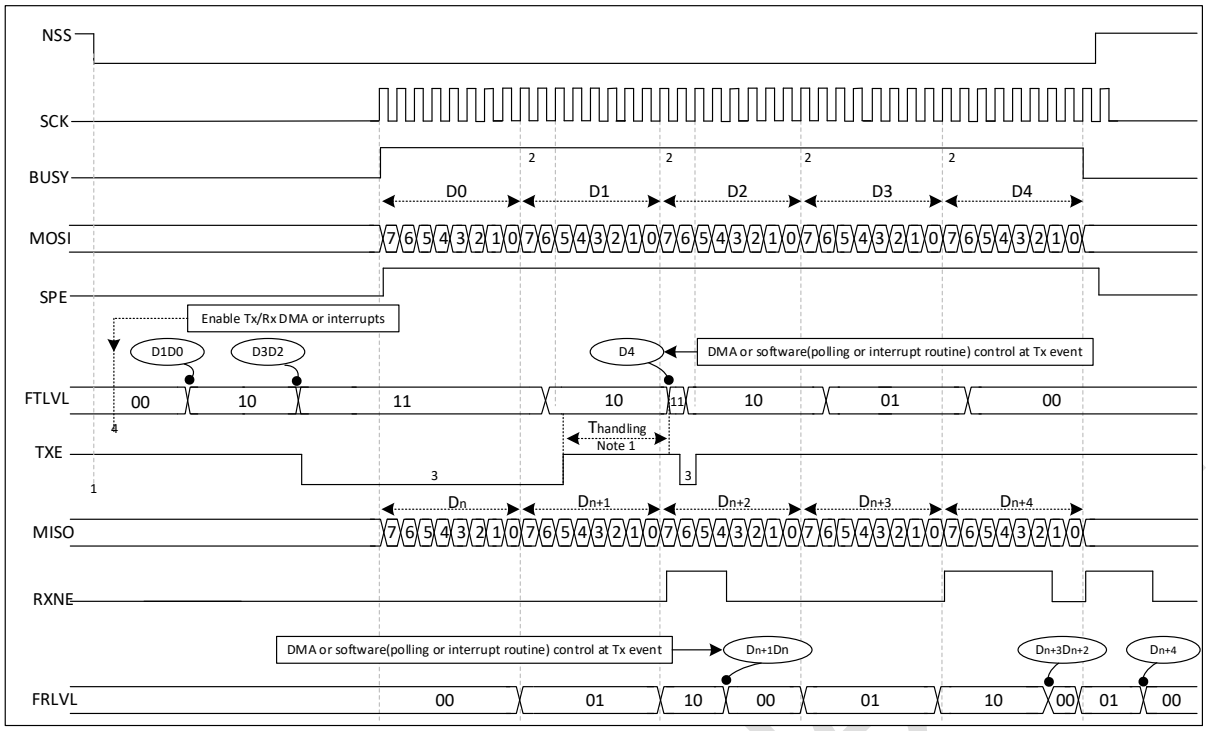


Figure 25-11 Host full-duplex communication timing diagram (bit frame=8)

25.2.10. Status Flag

The application can completely monitor the status of the SPI bus with 3 status flags.

25.2.10.1. Send buffer is empty flag (TXE)

The TXE flag bit is set when there is enough space in the TXFIFO to hold the data to be sent.The TXE flag bit is related to the TXFIFO level. This flag bit goes high and remains high until the TXFIFO level is less than or equal to 1/2 FIFO depth. If TXEIE (SPI_CR2) is set, an interrupt request is generated. When TXFIFO level is greater than 1/2, this bit is automatically cleared to zero.

25.2.10.2. Receive buffer non-empty flag (RXNE)

The RXNE flag bit is set:

- RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/4 (8-bit).

If the RXNEIE bit (SPI_CR2) is set, an interrupt is generated.

When the above condition no longer holds, RXNE is automatically cleared by hardware.

25.2.10.3. Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing this bit has no effect), and this flag indicates the state of the SPI communication layer.

When it is set to '1', it indicates that the SPI is busy communicating, with one exception: in the bidirectional receive mode of the master mode (MSTR=1, BDM=1 and BDOE=0), the BSY flag remains low during receive.

The BSY flag can be used to detect the end of a transmission before the software wants to shut down the SPI module and enter shutdown mode (or turn off the device clock), which will avoid corrupting the last transmission and therefore requires the following procedure to be strictly followed.

The BSY flag can also be used to avoid write conflicts in a multi-master system.

The BSY flag is set to '1' at the beginning of a transmission, except for the bidirectional receive mode of master mode (MSTR=1, BDM=1 and BDOE=0).

This flag is cleared to '0' in the following cases:

- When SPI is properly disable off
- Host mode, when MODF=1 is generated
- Host mode, when the transfer is complete and no more valid data is to be sent
- Slave mode, between each data transfer, the BSY flag is set to 0 and held for at least one SPI clock cycle

Note: Do not use the BSY flag for every data sent and received. It is more appropriate to use TXE and RXNE.

25.2.11. Error flags

25.2.11.1. Master Mode Failure (MODF)

Master Mode Failure (MODF) occurs only: when NSS is used as an input signal (SSOE=0), when the NSS pin of the master device is pulled low under NSS pin hardware mode management; or when the SSI bit is set to '0' under NSS pin software mode management. In this case, the MODF bit is automatically set. Master mode failure has the following effects on the SPI device:

- The MODF bit is set to '1' and an SPI interrupt is generated if the ERRIE bit is set;
- The SPE bit is cleared to '0'. This stops all outputs and shuts down the SPI interface;
- The MSTR bit is cleared to '0', thus forcing this device into slave mode.

The following steps are used to clear the MODF bit:

- Perform a read or write operation to the SPI_SR register when the MODF bit is set to '1';
- Then the SPI_CR1 register is written.

In systems with multiple MCUs, in order to avoid conflicts of multiple slave devices, the NSS pin of that master device must be pulled high before the MODF bit is cleared. After the clearing is completed, the SPE and MSTR bits can be restored to their original state.

For security reasons, the hardware does not allow the SPE and MSTR bits to be set when the MODF bit is '1'.

In normal configurations, the MODF bit cannot be set to '1' for slave devices. However, in a multi-master configuration, a device can be in slave mode with the MODF bit set; in this case, the MODF bit indicates a possible multi-master conflict. The interrupter can perform a reset or return to the default state to recover from the error state.

25.2.11.2. Overrun flag (OVR)

The overrun situation occurs when data is received by the host or slave and there is not enough space in the RXFIFO to store the received data. This happens if the software or DMA does not have enough time to read away the previously received data (stored in the RXFIFO).

When the overrun condition occurs, the newly received data does not overwrite the data previously stored in the RXFIFO. The new data received is ignored and all next data sent is lost.

The OVR can be cleared by reading the SPI_DR register and the SPI_SR register in sequence.

25.2.12. SPI Interrupt

Table 25-1 SPI Interrupt request

Interrupt event	Event Flag	Enable control bit
TXFIFO waiting to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master mode failure event	MODF	ERRIE
Overflow error	OVR	ERRIE

25.2.13. CRC Calculation

To check the reliability of the transmitted and received data, two independent CRC calculators are utilized. The SPI provides a CRC8 or CRC16 calculation independent of the frame data length, which can be fixed to 8 or 16 bits. For all other data frame lengths, no CRC is available.

25.2.13.1. CRC Guidelines

Before SPI is enabled (SPE = 1), CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register. The CRC value is calculated using an odd number of programmable polynomials on each bit. This calculation is processed on the sample clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is automatically checked at the end of the data block and its transfer is managed by the CPU or DMA. The CRCERR flag is set to indicate a data error when a mismatch is detected between the CRC calculated internally based on the received data and the CRC sent by the transmitter. The correct procedure for handling CRC calculations depends on the SPI configuration and the selected transfer management method.

25.2.13.2. CPU managed CRC transfer

Begin and communicate continuously until the last data frame in the SPIx_DR register is sent or received. The CRCNEXT bit must then be set in the SPIx_CR1 register to indicate that transmission of CRC frames begins after the currently processed data frame. The CRCNEXT bit must be set before the last data frame transmission is completed. Stops CRC calculation during CRC transmission.

The received CRC data is stored in the RXFIFO as a byte or word. This is why in CRC mode only, the receive buffer must be treated as a single 16-bit buffer for receiving only one data frame at a time. a CRC format transmission usually requires one more data frame at the end of the data transmission. However, when setting up an 8-bit data frame that passes the 16-bit CRC checksum, two more frames are required to send the full CRC value.

When the last CRC data is received, an automatic checksum is performed to compare the received value with the SPIx_RXCRC register. The software must check the CRCERR flag in the SPIx_SR register to determine if there is an error in the data transmission. After CRC reception, the CRC value is stored in the RXFIFO and the SPIx_DR register must be read to clear the RXNE flag.

25.2.13.3. DMA managed CRC transfer

When SPI communication is enabled using DMA mode with CRC, CRC transmission and reception at the end of communication is done automatically (except for reading CRC data in receive-only mode) and the CRCNEXT bit does not have to be handled by software. The counter

for the SPI transfer DMA channel must be set to the number of data frames to be transferred, which does not include CRC frames. On the receiver side, the received CRC value is automatically handled by the DMA at the end of the transmission, but the user must take care to clear the received CRC information from the RXFIFO, as it is always stored in the RXFIFO. In full duplex mode, the counter of the receive DMA channel can be set to the number of data frames to be received including the CRC.

In receive-only mode, the DMA receive channel counter should contain only the amount of data transferred and not the CRC calculation. Then the complete DMA-based transfer, since it works as a single buffer in this mode, all CRC values must be read back from the FIFO by software. At the end of the data and CRC transfer, the CRCERR flag in the SPIx_SR register is set if an error occurs during the transfer.

25.3. I2S Function Description

25.3.1. Introduction

The I2S function can be enabled by setting the I2SMOD position of register SPI_I2SCFGR to '1'. At this point, the SPI module can be used as an I2S audio interface. The I2S interface uses approximately the same pins, flags and interrupts as the SPI interface.

I2S and SPI share 3 pins:

SD: serial data (mapped to MOSI pin), which is used to send and receive data from the 2-way time division multiplexed channel;

WS: word select (mapped to the NSS pin), used as a data control signal output in master mode and as an input in slave mode;

CK: Serial clock (mapped to the SCK pin), output as clock signal in master mode, input in slave mode.

An additional pin can be used to output the clock when a master clock is required for certain external audio devices:

MCK: Master clock (independently mapped), used as an additional clock signal pin for output when I2S is configured in master mode and the MCKOE bit of register SPI_I2SPR is 'for K. The frequency of the output clock signal is pre-set to 256 × Fs, where Fs is the sampling frequency of the audio signal.

When set to master mode, the I2S uses its own clock generator to generate the clock signal for communication. This clock generator is also the clock source for the master clock output. There are 2 additional registers in I2S mode, one is the register SPI_I2SPR related to clock generator configuration and the other is the I2S general configuration register SPI_I2SCFGR (which can set parameters such as audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPI_CR1 register and all CRC registers are not used in I2S mode. Similarly, the SSOE bit of register SPI_CR2, and the MODF and CRCERR bits of register SPI_SR are not used in I2S mode.

I2S uses the same register SPI_DR as SPI for 16-bit wide mode data transfer.

25.3.2. Audio Protocol

The 3-wire bus supports time-division multiplexing of audio data on 2 channels: the left channel and the right channel, but only one 16-bit register is used for transmitting or receiving. Therefore, when writing data to the data register, the software must write the corresponding data according to the channel currently being transmitted; similarly, when reading the register data, the CHSIDE bit of register SPI_SR is checked to determine which channel the received data belongs to. The left channel always sends data before the right channel (the CHSIDE bit is meaningless under the PCM protocol). There are four available combinations of data and packet frames. Data can be sent in the following four data formats:

- Have 16-bit data packed into 16-bit frames
- Frame with 16-bit data packed into a 32-bit frame
- Frame with 24-bit data packed into a 32-bit frame
- Frame 32-bit data packed into 32-bit frame

When using 16-bit data extension to 32-bit frames, the first 16 bits (MSB) are meaningful data and the last 16 bits (LSB) are forced to 0. This operation requires no software intervention and no DMA request (only one read/write operation is required). 24-bit and 32-bit data frames require 2 CPU read or write operations to register SPI_DR, and when using DMA, 2 DMA transfers. For 24-bit data, the lowest 8 bits are set to 0 by hardware after expansion to 32 bits. for all data formats and communication standards, the highest bit (MSB) is always sent first.

The I2S interface supports four audio standards, which can be selected by setting the I2SSTD[1:0] bits and PCMSYNC bits of register SPI_I2SCFGR.

25.3.2.1. I2S Philips protocol

In this standard, pin WS is used to indicate which channel the data being sent belongs to. This pin is active 1 clock cycle before the first data (MSB) is sent.

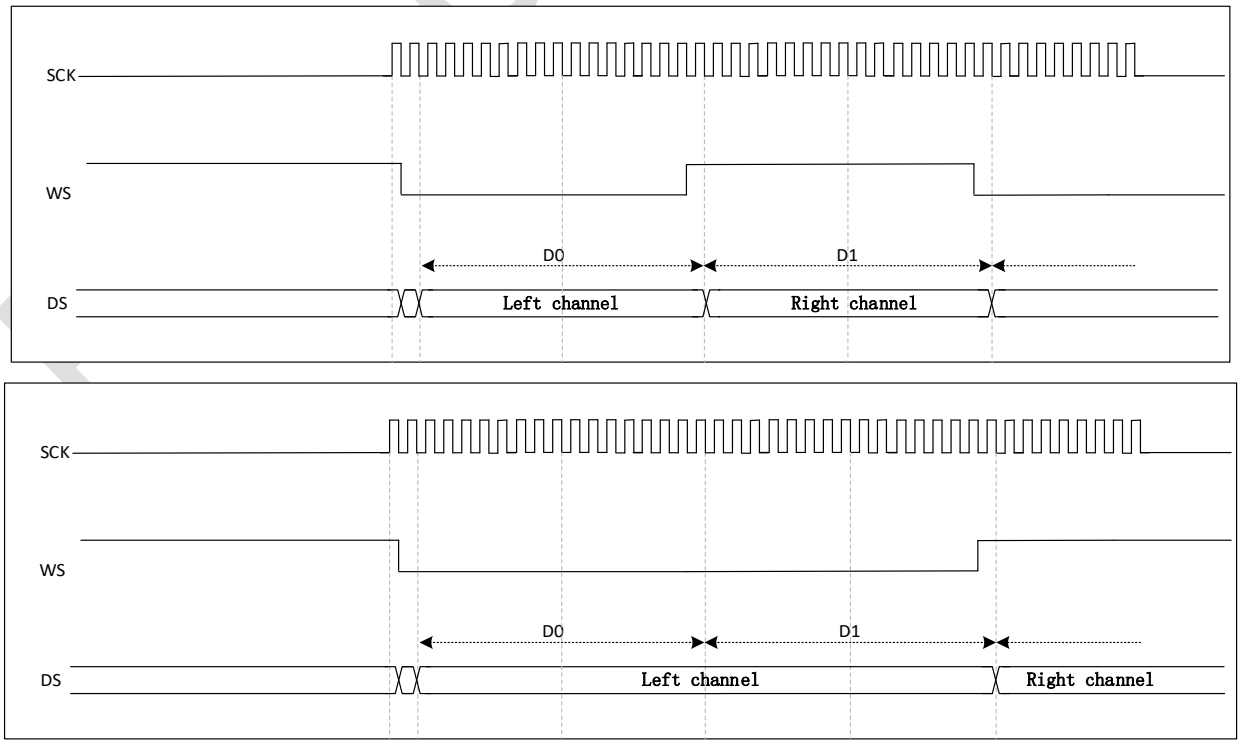


Figure 25-12 I2S Philips protocol waveform (16/32-bit full precision, CKPOL = 0)

When CKPOL=1 ,

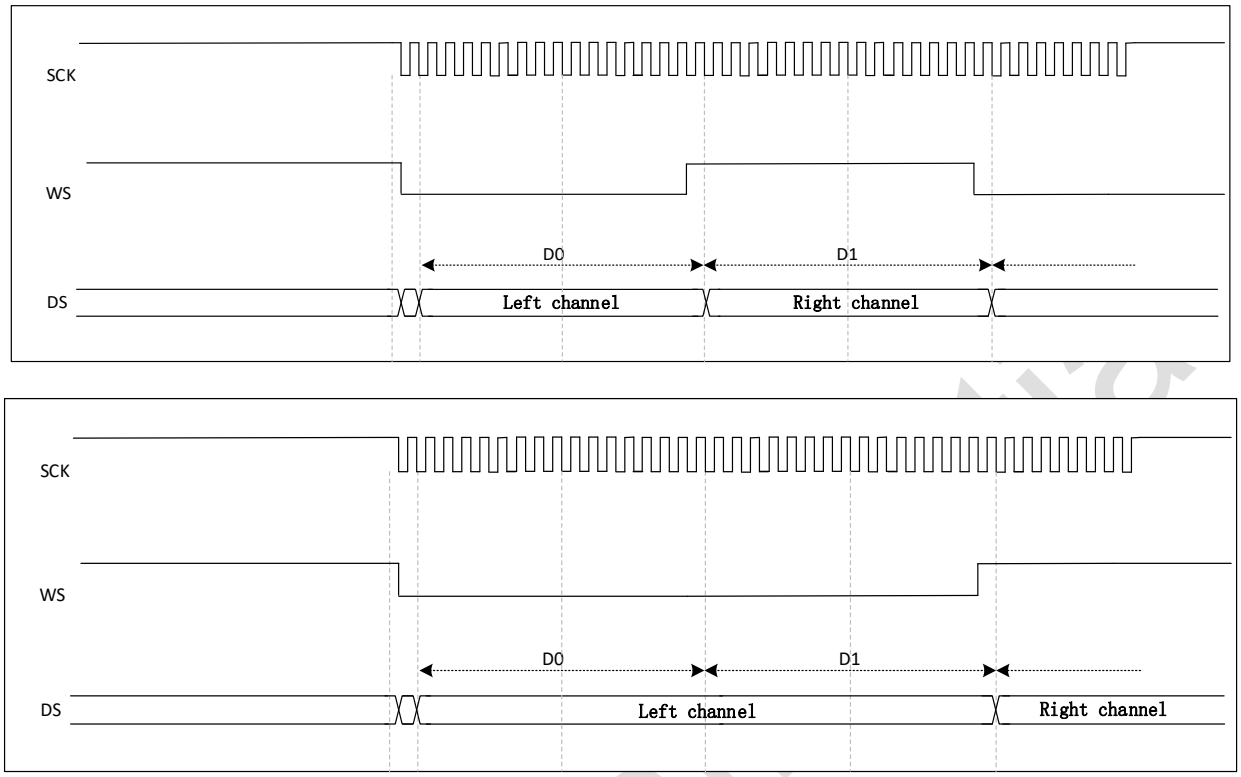


Figure 25-13 I2S Philips protocol waveform (16/32-bit full precision, CKPOL = 1)

The sender changes the data on the falling edge of the clock signal (CK) and the receiver reads the data on the rising edge. The WS signal also changes on the falling edge of the clock signal.

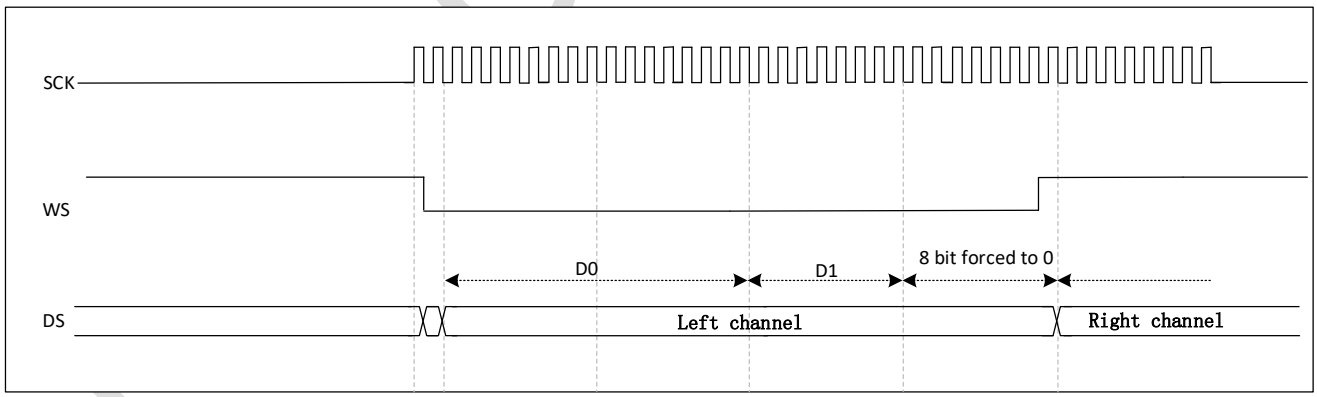


Figure 25-14 I2S Philips protocol standard waveform (24-bit frame, CKPOL = 0) When CKPOL = 1, the sender also changes the data on the falling edge of the clock signal (CK)

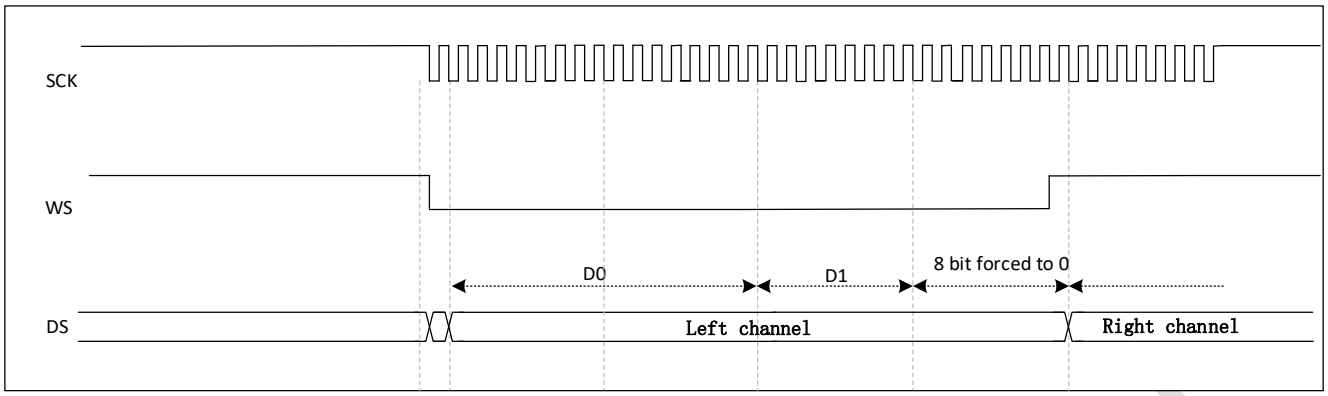


Figure 25-15 I2S Philips protocol standard waveform (24-bit frame, CKPOL = 1)

This mode requires 2 read or write operations to the register SPI_DR.

- Read in transmit mode: If you need to send 0x8EAA33 (24 bits):

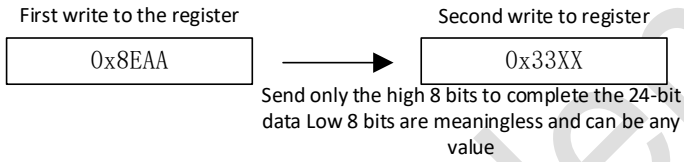


Figure 25-16 Read in send mode

Send 0x8EAA33

- In receive mode: If receiving 0x8EAA33:

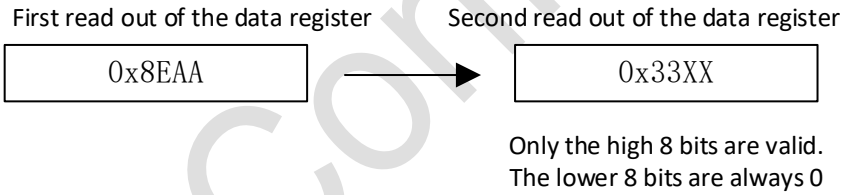


Figure 25-17 Read in receive mode

Receive 0x8EAA33

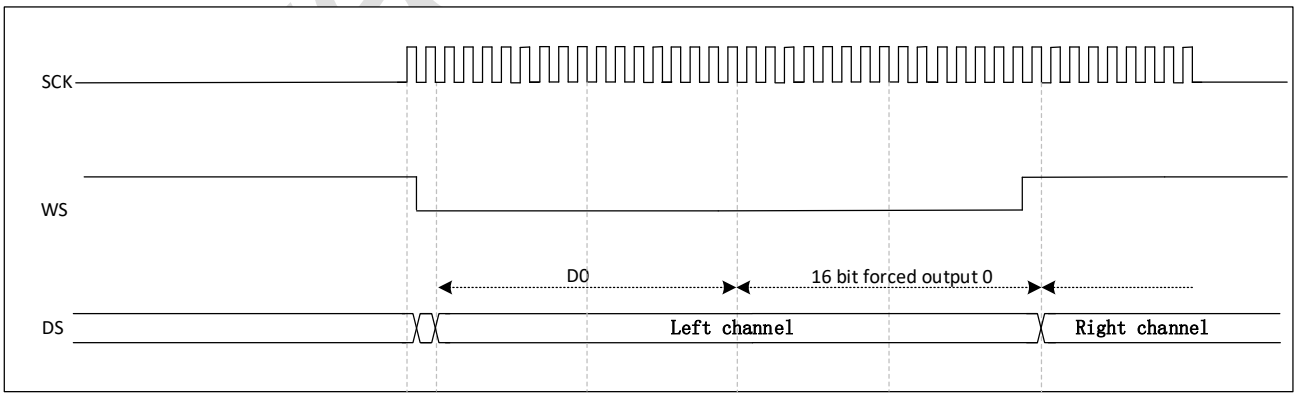


Figure 25-18 I2S Philips protocol standard waveform (16-bit extended to 32-bit packet frame, CKPOL = 0)

When CKPOL = 1, the sender also changes the data on the falling edge of the clock signal (CK)

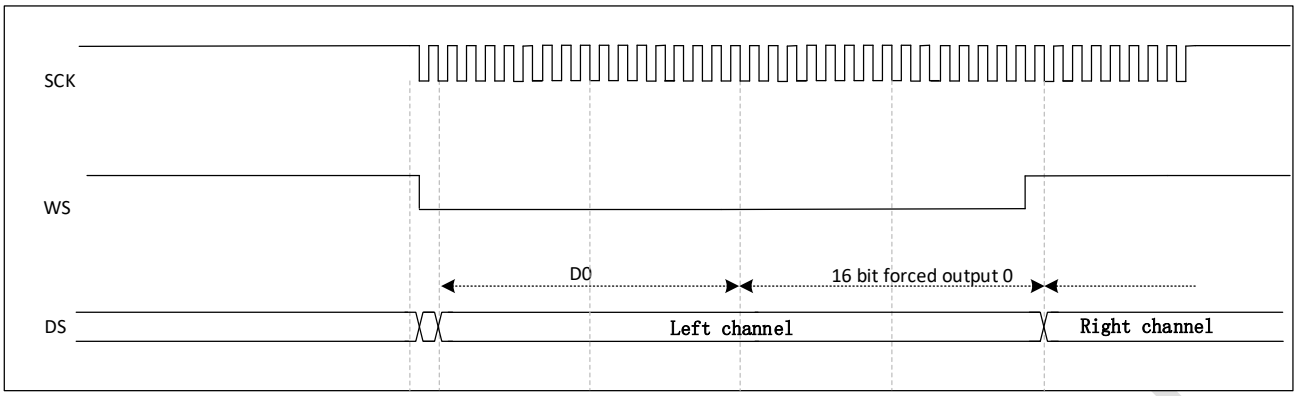


Figure 25-19 I2S Philips protocol standard waveform (16-bit extended to 32-bit packet frame, CKPOL = 1)

During the I2S configuration phase, if you choose to extend the 16-bit data to 32-bit channel frames, you only need to access the SPI_DR register once. the lower 16 bits used to extend to 32 bits are set to 0x0000 in hardware.

If the data to be sent or received is 0x76A3 (extended to 32 bits as 0x76A30000), perform only one operation on SPI_DR.

The MSB needs to be written to register SPI_DR when sending; flag bit TXE is "E" to indicate that new data can be written and an interrupt can be generated if the corresponding interrupt is allowed. Sending is done by hardware, and TXE is set and the corresponding interrupt is generated even if the last 16 bits of 0x0000 have not been sent yet. When receiving, each time a high 16-bit half word (MSB) is received, the flag bit RXNE is set to "N and an interrupt can be generated if the corresponding interrupt is allowed.

This allows more time between the 2 reads and writes and prevents underflow or overflow.

25.3.2.2. MSB Alignment Standard

In this standard, the WS signal and the first data bit, the highest bit (MSB), are generated at the same time.

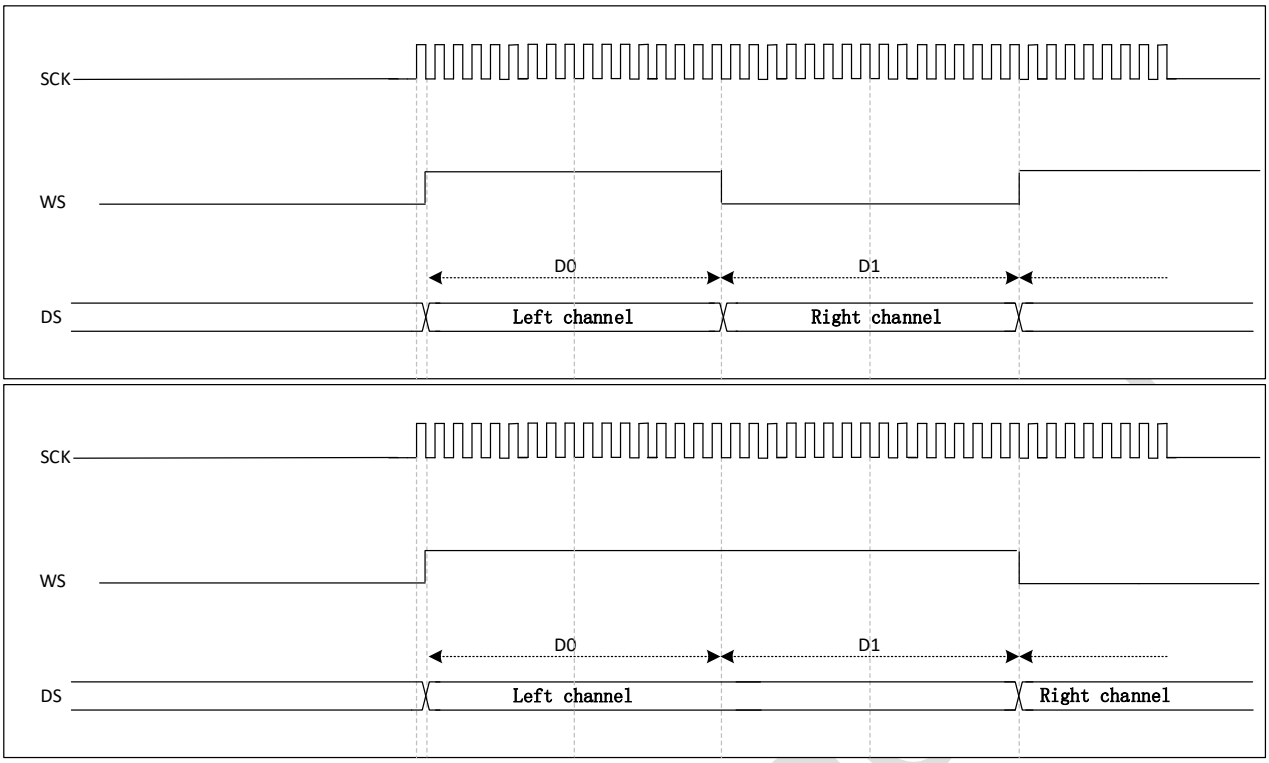


Figure 25-20 MSB aligned 16-bit or 32-bit full precision, CKPOL = 0

When CKPOL = 1, the sender also changes the data on the falling edge of the clock signal (CK)

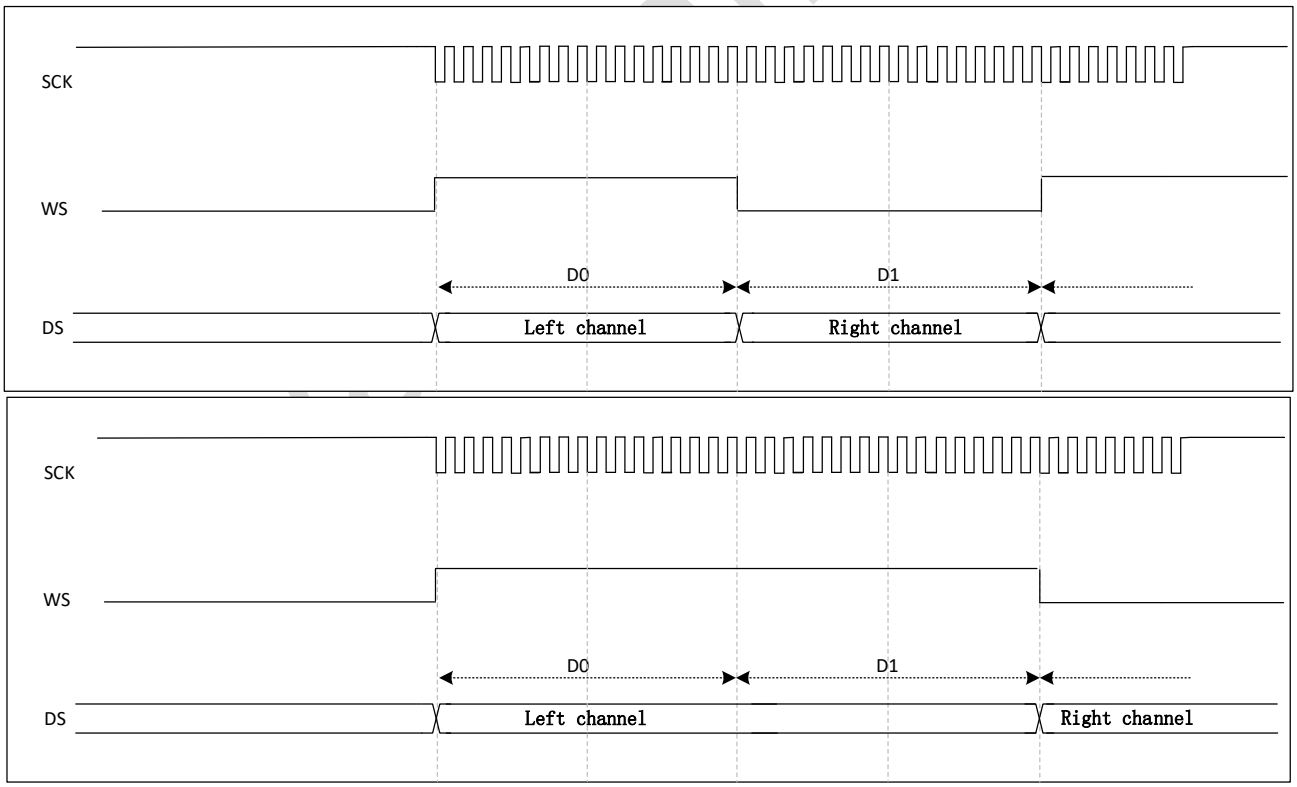


Figure 25-21 MSB aligned 16-bit or 32-bit full precision, CKPOL = 1

The sender changes the data on the falling edge of the clock signal; the receiver is reading the data on the rising edge.

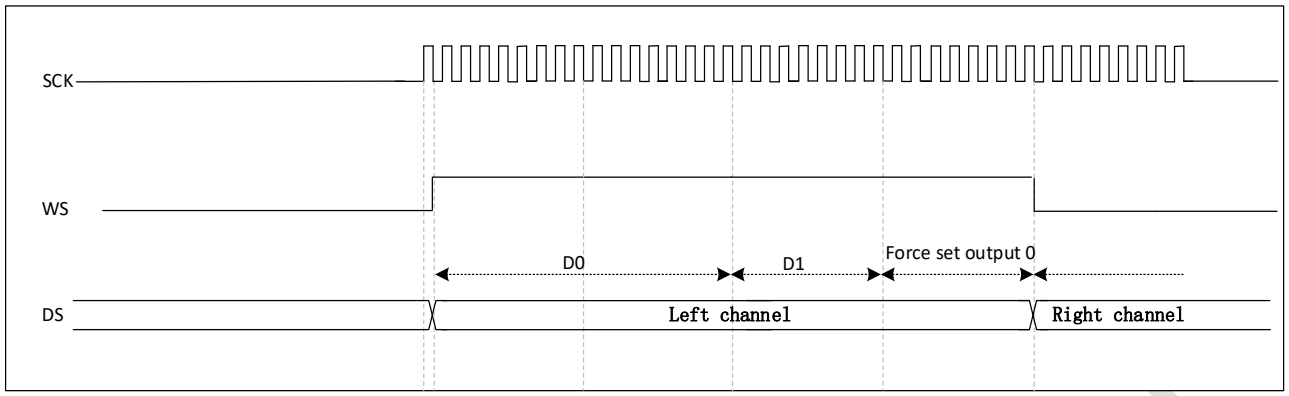


Figure 25-22 MSB aligned 24-bit data, CKPOL = 0

When CKPOL = 1, the sender also changes the data on the falling edge of the clock signal (CK)

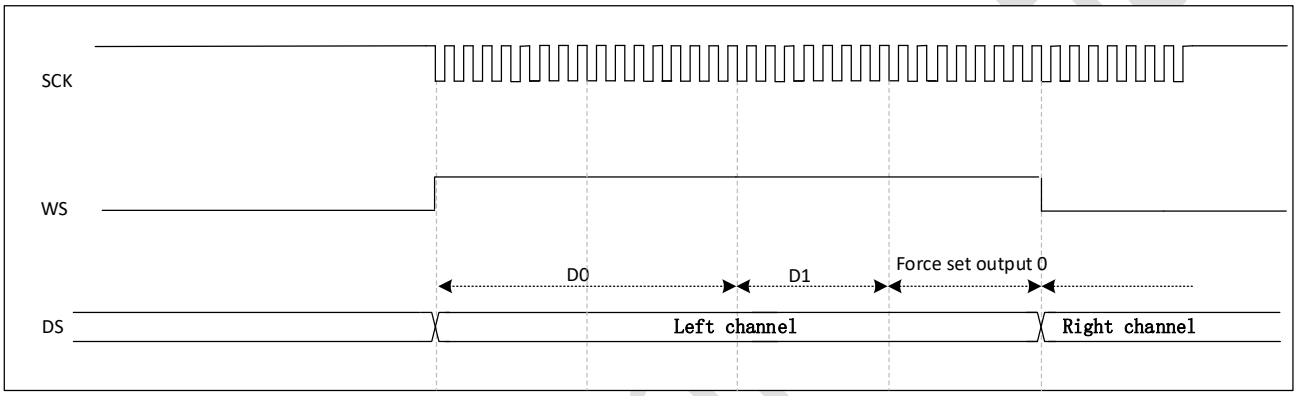


Figure 25-23 MSB aligned 24-bit data, CKPOL = 1

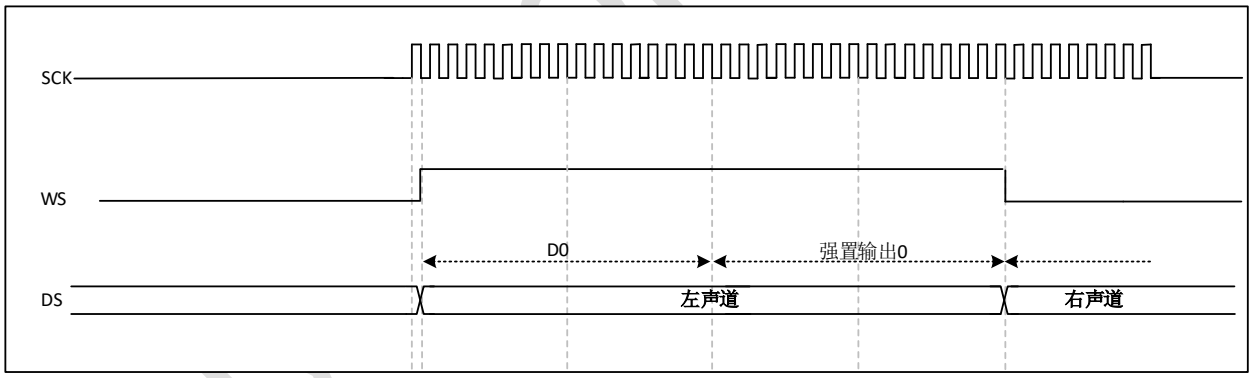


Figure 25-24 MSB aligned 16-bit data extension to 32-bit packet frame, CKPOL = 0

When CKPOL = 1, the sender also changes the data on the falling edge of the clock signal (CK)

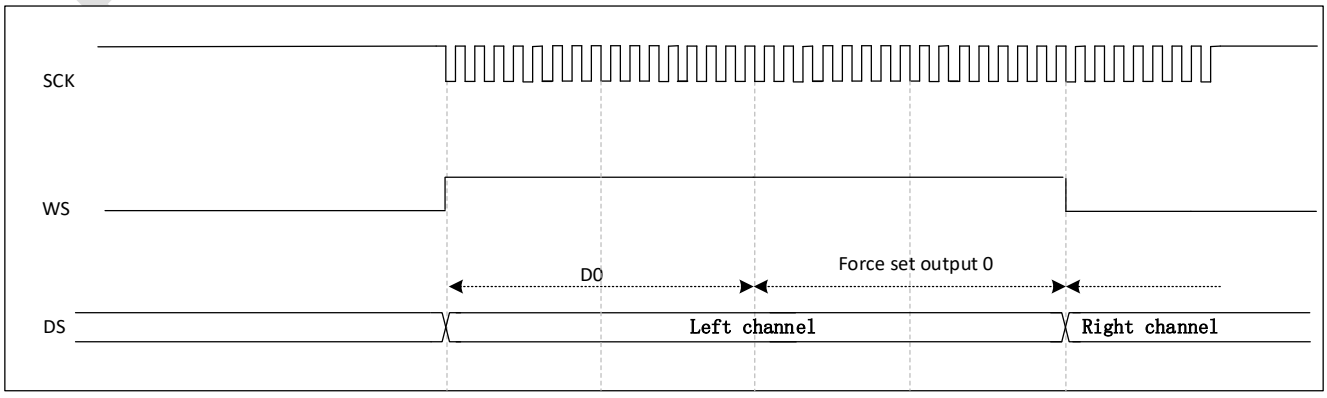


Figure 25-25 MSB aligned 16-bit data extension to 32-bit packet frame, CKPOL = 1

The next TXE event occurs as soon as valid data starts to be sent out from the SD pin. On receive, the RXNE event occurs as soon as valid data is received (not the 0x0000 part).

25.3.2.3. LSB alignment standard

This standard is similar to the MSB alignment standard (no difference in 16-bit or 32-bit full-precision frame formats).

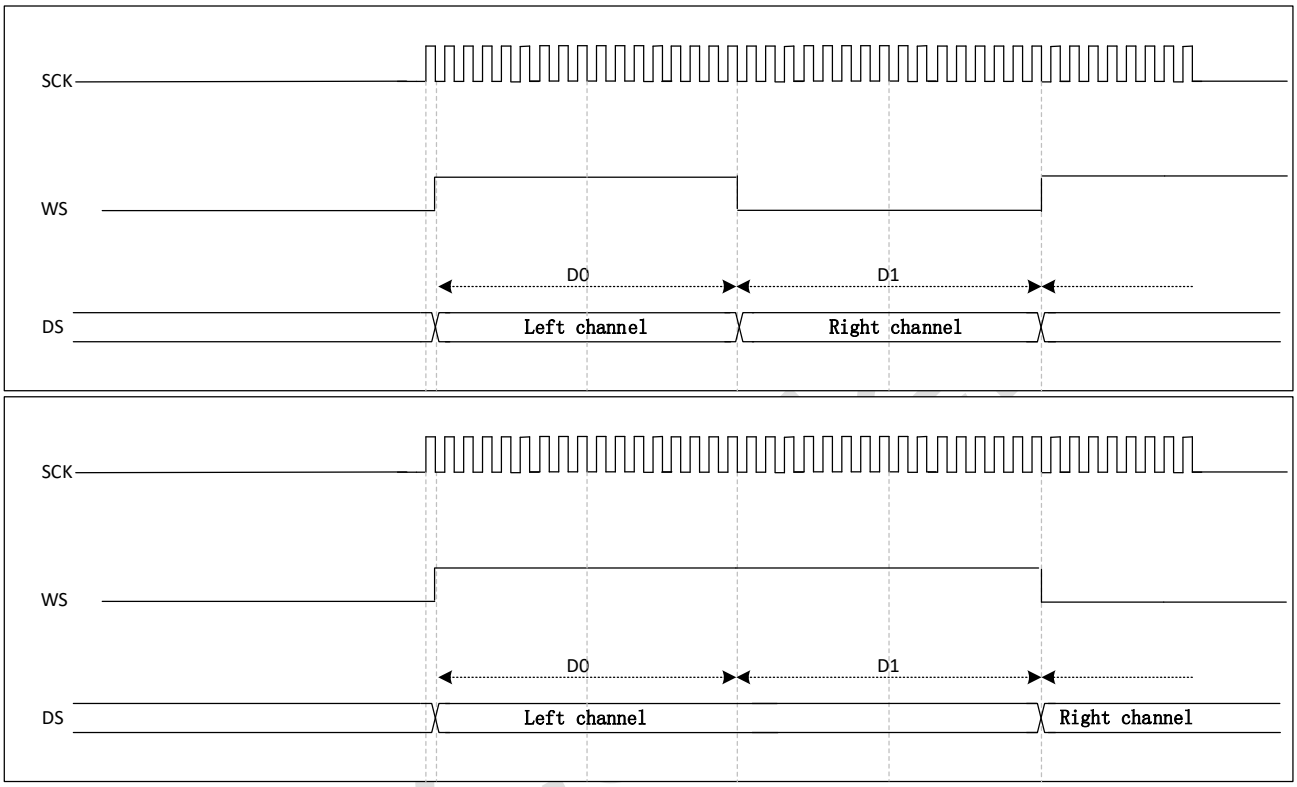


Figure 25-26 LSB aligned 16-bit or 32-bit full precision, CKPOL = 0

When CKPOL = 1, the sender also changes the data on the falling edge of the clock signal (CK)

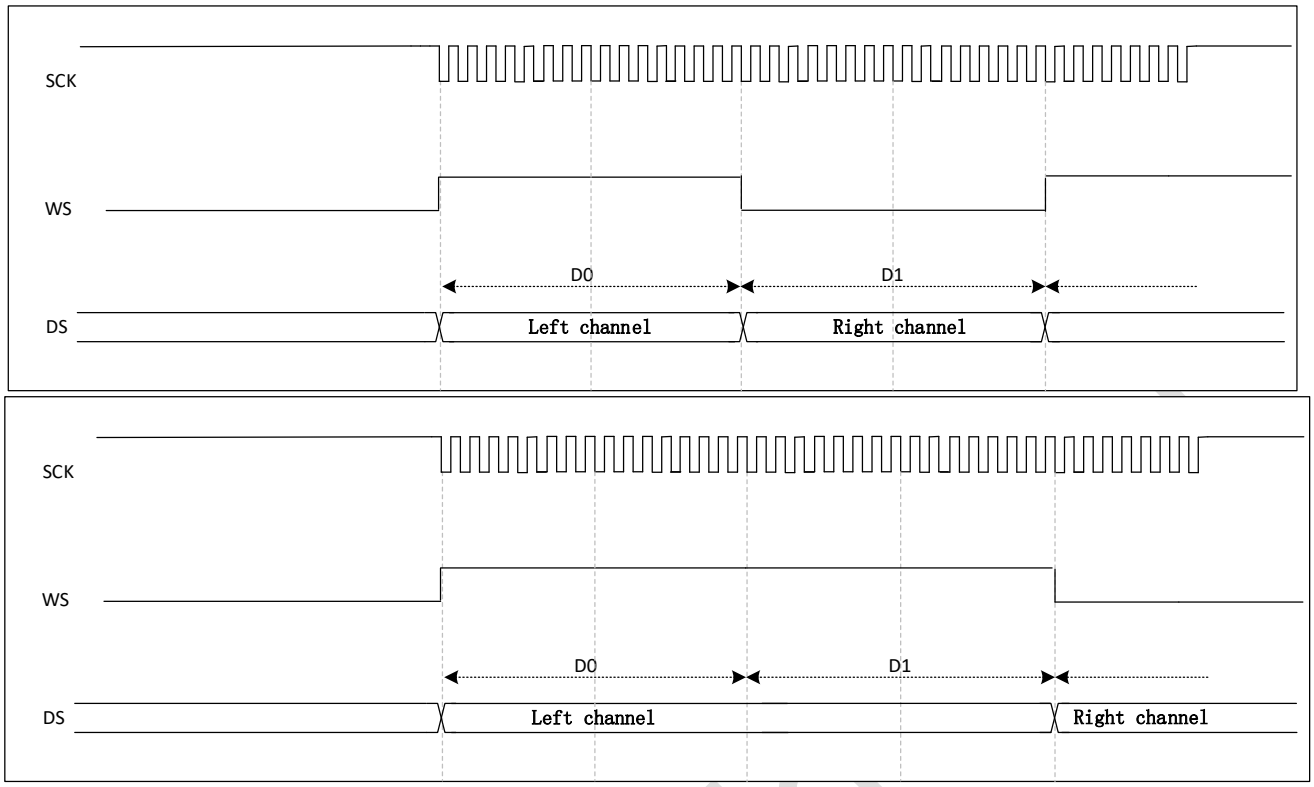


Figure 25-27 LSB aligned 16-bit or 32-bit full precision, CKPOL = 1

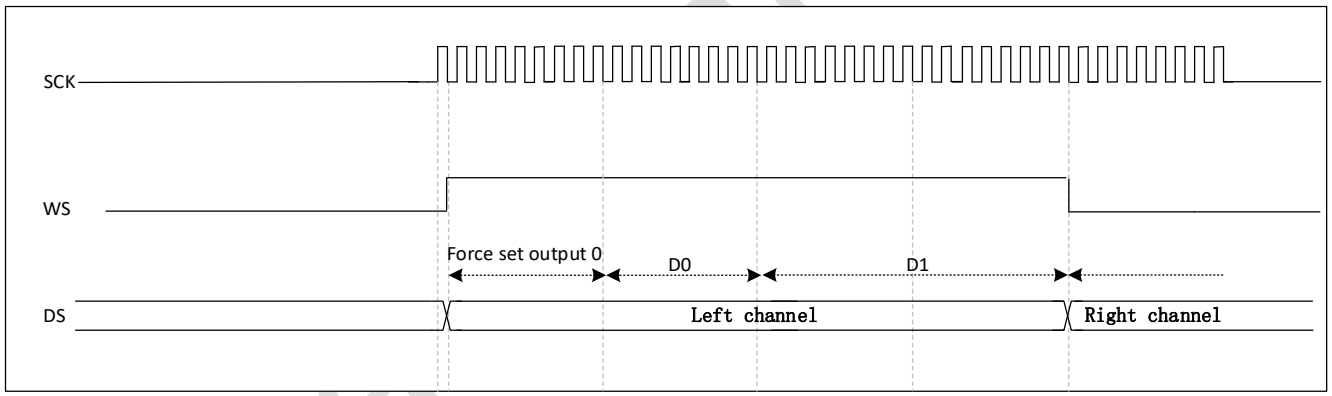


Figure 25-28 LSB aligned 24-bit data, CKPOL = 0

When CKPOL = 1, the sender also changes the data on the falling edge of the clock signal (CK)

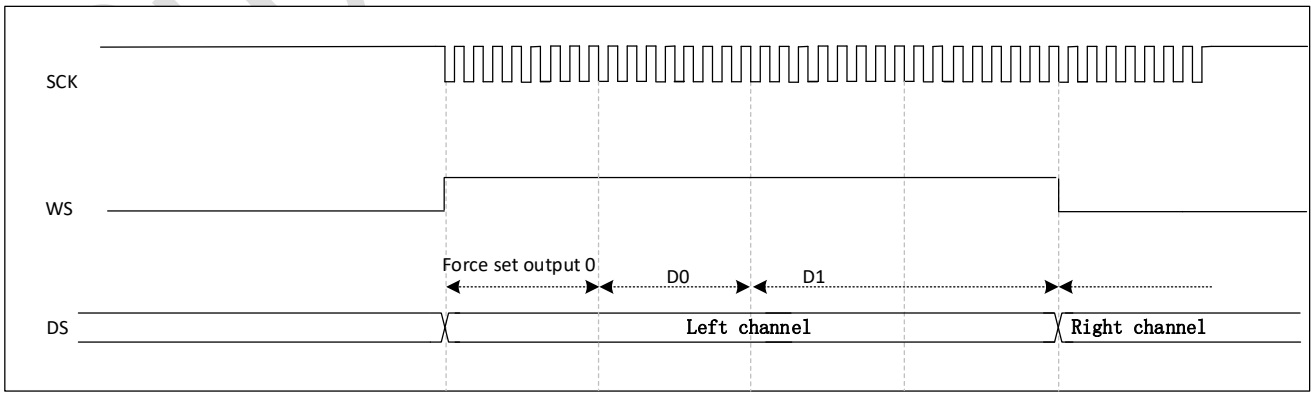


Figure 25-29 LSB aligned 24-bit data, CKPOL = 1

- In send mode

To send data 0x3478AE, 2 write operations to register SPI_DR are required by software or DMA. The operation flow is shown in the figure below.

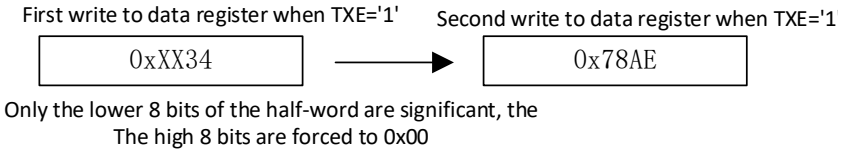


Figure 25-30 In send mode

Request to send the operation 0x3478AE

- In receive mode

To receive data 0x3478AE, it is necessary to perform 1 read operation on each of the 2 consecutive RXNE events to register SPI_DR.

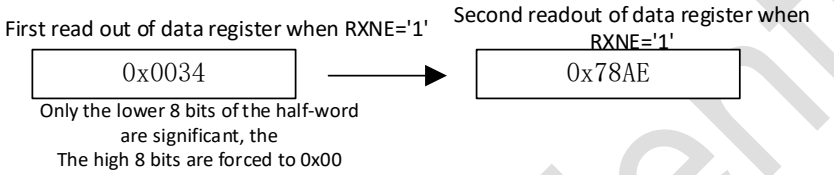


Figure 25-31 In receive mode

Request to receive the operation 0x3478AE

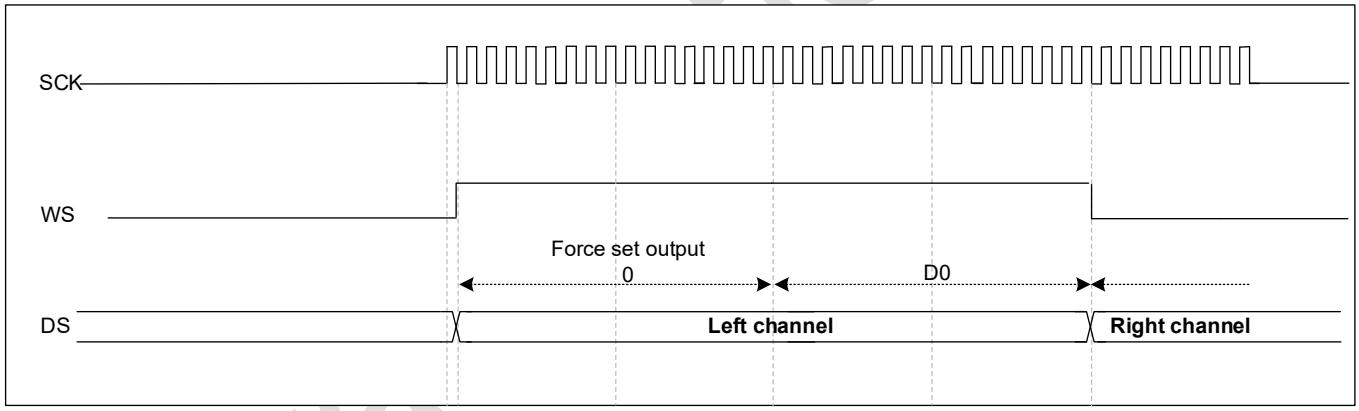


Figure 25-32 LSB aligned 16-bit data extension to 32-bit packet frame, CKPOL = 0

When CKPOL = 1, the sender also changes the data on the falling edge of the clock signal (CK)

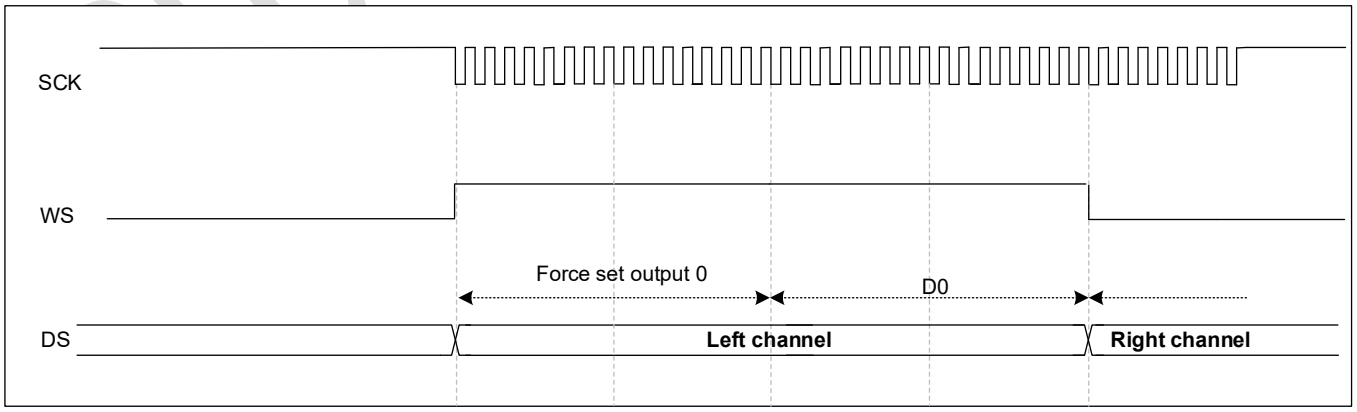


Figure 25-33 LSB aligned 16-bit data extension to 32-bit packet frame, CKPOL = 1

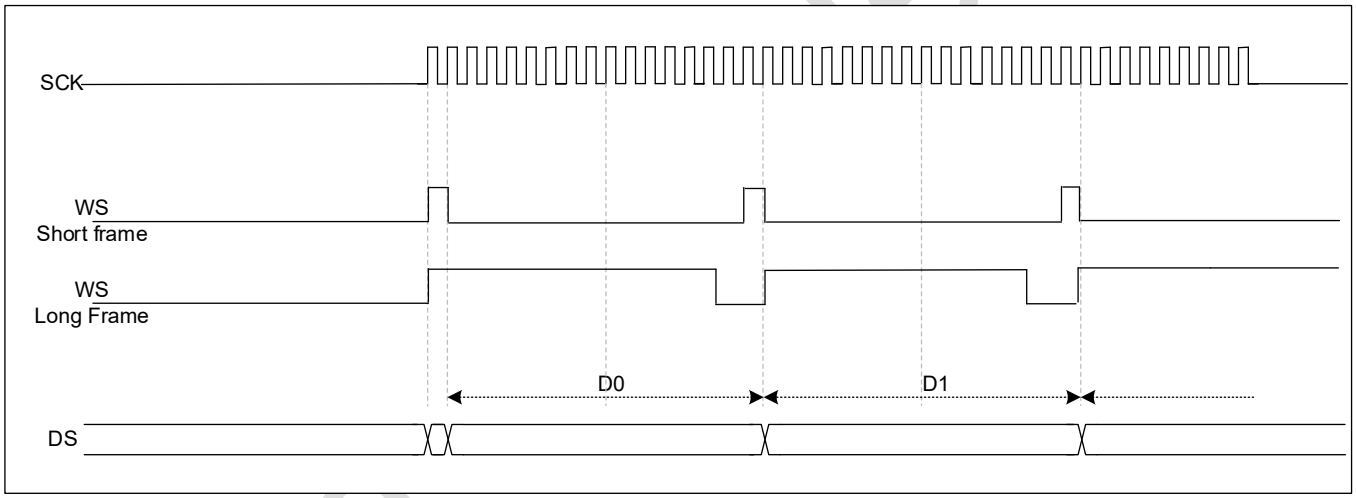
During the I2S configuration phase, if you choose to extend the 16-bit data to a 32-channel frame, you only need to access the SPI_DR register once. At this point, the high half word (16-bit MSB) after the extension to 32 bits is set to 0x0000 by hardware.

If the data to be sent or received is 0x76A3 (extended to 32 bits as 0x0000 76A3), operate SPI_DR only once.

When sending, if TXE is "E, the user needs to write the data to be sent (i.e. 0x76A3). The 0x0000, used to extend to 32 bits, is partially sent out by hardware first, and the next TXE event occurs once valid data starts to be sent out from the SD pin. On receive, the RXNE event occurs as soon as valid data is received (not the 0x0000 part). This allows more time between the 2 reads and writes and prevents underflow or overflow from occurring.

25.3.2.4. PCM Standard

With the PCM standard, there is no information about the sound channel selection. The PCM standard has 2 available frame structures, short frame or long frame, which can be selected by setting the PCMSYNC bit of register SPI_I2SCFGR.



When CKPOL = 1, the sender also changes the data on the rising edge of the clock signal (CK)

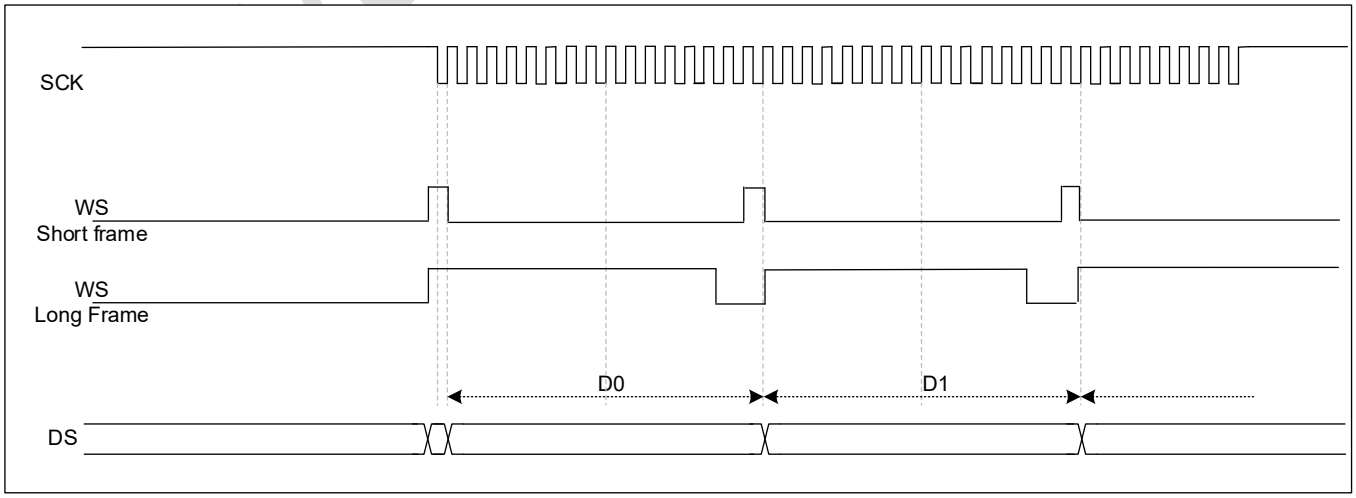
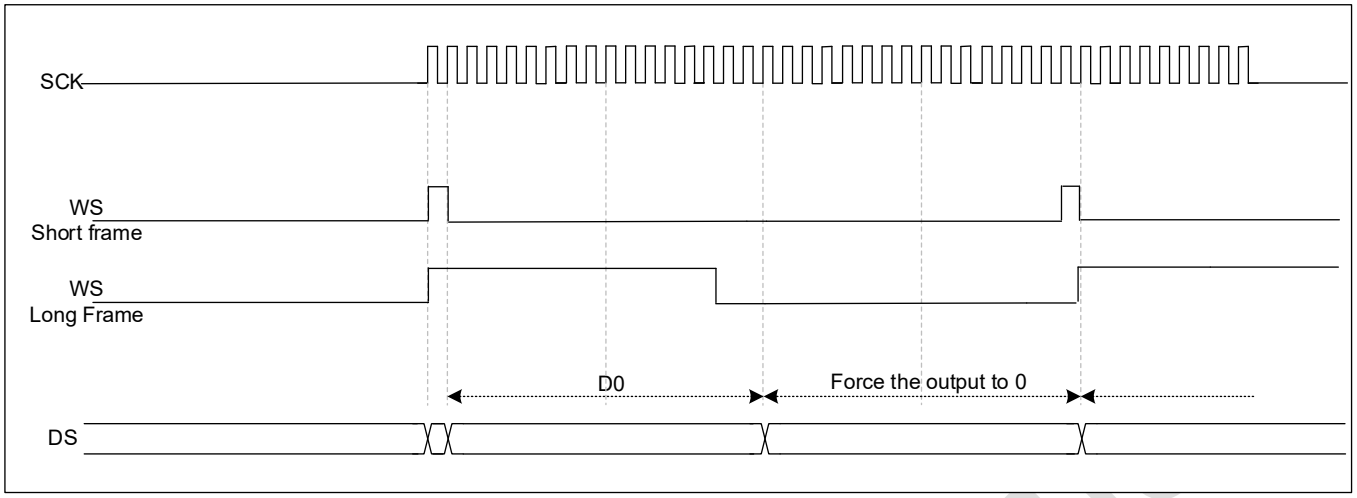


Figure 25-34 PCM standard waveform (16 bits)

For long frames, the WS signal used for synchronization in master mode is valid for a fixed duration of 13 bits.

For short frames, the WS signal used to synchronize is only 1 bit long.



When CKPOL = 1, the sender also changes the data on the rising edge of the clock signal (CK)

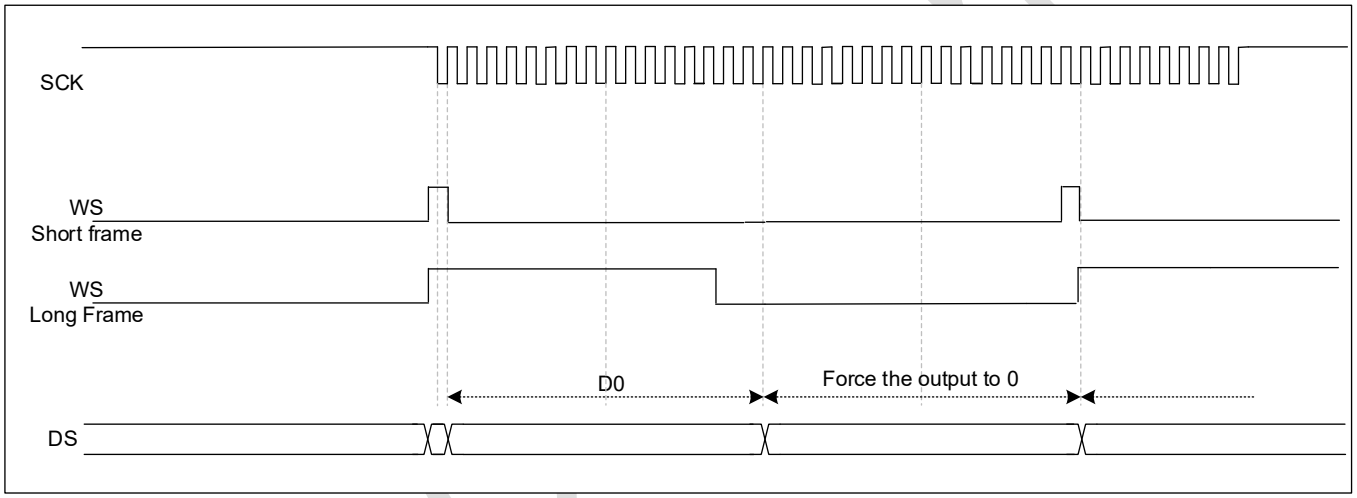


Figure 25-35 PCM standard waveforms (16-bit extended to 32-bit packet frames)

Regardless of the mode (master or slave) and the synchronization method (short or long frame), the time difference between 2 consecutive frames of data and between 2 synchronization signals (even in slave mode) needs to be determined by setting the DATLEN and CHLEN bits of the SPI_I2SCFGR register.

25.3.3. Clock

The I2S bit rate i.e. determines the data flow on the I2S data line and the frequency of the I2S clock signal. That is, I2S bit rate = number of bits per channel x number of channels x audio sampling frequency.

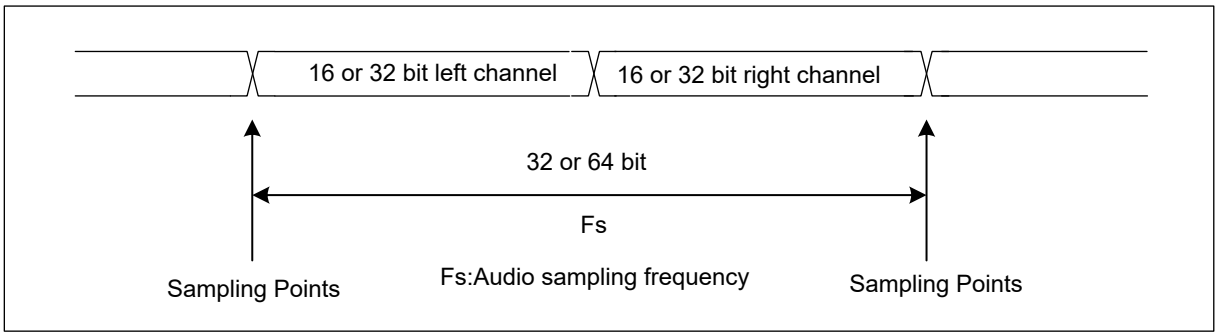
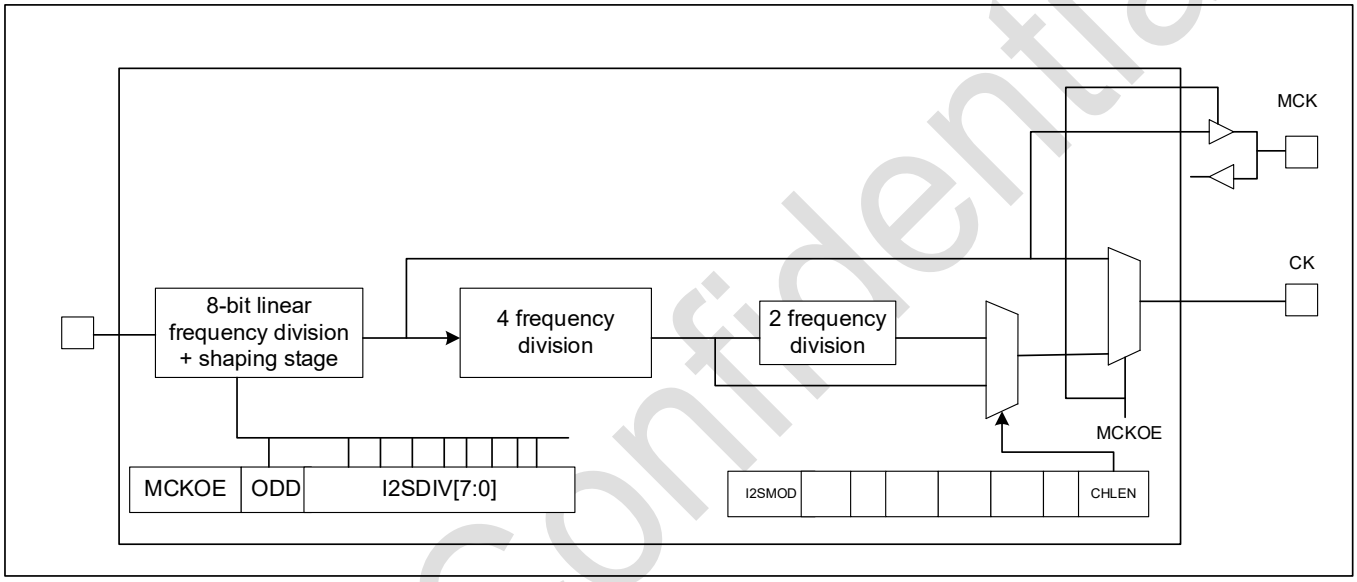


Figure 25-36 Audio Sample Definition

In master mode, the linear divider needs to be set correctly in order to obtain the desired audio frequency.

I2S clock generator architecture



The audio sampling frequency can be 96kHz, 48kHz, 44.1kHz, 32kHz, 22.05kHz, 16kHz, 11.025kHz or 8kHz (or any value in this range). To obtain the desired frequency, set the linear divider according to the following formula, when the master clock needs to be generated (the MCKOE bit of the SPI_I2SPR register is 'for KO:

When the frame length of the sound channel is 16 bits, Fs=I2SxCLK/[(16*2)*((2*I2SDIV)+ODD)*8]

When the frame length of the sound channel is 32 bits, Fs=I2SxCLK/[(32*2)*((2*I2SDIV)+ODD)*4]

When the master clock is turned off (MCKOE bit is 'for' K:

When the frame length of the sound channel is 16 bits, Fs=I2SxCLK/[(16*2)*((2*I2SDIV)+ODD)]

When the frame length of the sound channel is 32 bits, Fs=I2SxCLK/[(32*2)*((2*I2SDIV)+ODD)]

25.3.4. Send and Receive

25.3.4.1. Master Mode

Set the I2S to operate in master mode, with the serial clock output from pin CK and the word select signal generated from pin WS. You can choose to output or not to output the master clock (MCK) by setting the MCKOE bit of register SPI_I2SPR.

- Sending process

The transmit process starts when 1 half-word (16 bits) of data is written to the transmit buffer.

It is assumed that the first data written to the transmit buffer corresponds to the left channel data.

When the data is moved from the transmit buffer to the shift register, the flag bit TXE is set to '1', at

which point the data corresponding to the right channel is to be written to the transmit buffer. The flag bit CHSIDE indicates which channel corresponds to the data currently to be transferred. The value of the flag bit CHSIDE is updated when TXE is '1', so it has meaning when TXE is '1'. A complete data frame is not considered until all data from the left channel first and then the right channel are transmitted. It is not possible to transmit only part of the data frame, e.g. data for the left channel only.

While the first bit of data is sent, the half-word data is transferred to the 16-bit shift register in parallel, and then the subsequent bits are sent from pin MOSI/SD in order of high bit first. Each time the data is moved from the transmit buffer to the shift register, the flag bit TXE is set to '1', and an interrupt is generated if the TXEIE bit of register SPI_CR2 is '1'. To ensure continuous audio data transmission, it is recommended to write the next data to be transmitted to register SPI_DR before the current transmission is completed.

- Receiving Process

Audio data is always received in 16-bit packets, regardless of data and channel length. That is, the flag bit RXNE is set to '1' each time the receive buffer is filled, and an interrupt is generated if the RXNEIE bit in register SPI_CR2 is '1'. Depending on the configured data and channel length, receiving data for the left or right channel will require one or two transfers of data to the receive buffer. The RXNE flag bit is cleared by reading the SPI_DR register. The CHSIDE is updated after each reception and its value depends on the WS signal generated by the I2S unit.

If the previous received data has not been read and new data is received, i.e., an overflow occurs, the flag bit OVR is set to '1', and if the ERRIE bit of register SPI_CR2 is '1', an interrupt is generated, indicating that an error has occurred . To turn off the I2S function, special operations need to be performed to ensure that the I2S module can complete the transfer cycle normally without starting a new data transfer; the BSY flag is always low during the transfer.

25.3.4.2. Slave Mode

In slave mode, I2S can be set to transmit and receive modes. The configuration of the slave mode follows basically the same procedure as the configuration of the master mode. In slave mode, no clock is required from the I2S interface. Both the clock signal and the WS signal are provided by the external master I2S device, connected to the corresponding pins. Therefore the user does not need to configure the clock.

- Sending process

The transmit process starts when the external master device sends the clock signal and when the NSS_WS signal requests data transfer. The slave device must be enabled and the I2S data register must be written before the external master device can start communication.

For I2S MSB-aligned and LSB-aligned modes, the first data item written to the data register corresponds to the left channel data. When communication starts, data is transferred from the transmit buffer to the shift register and then the flag bit TXE is set to '1'; at this point, the data item corresponding to the right channel is to be written to the I2S data register. The flag bit CHSIDE indicates which channel corresponds to the data currently to be transmitted. In contrast to the transmit process in master mode, in slave mode CHSIDE depends on the WS signal from the external master I2S. This means that the slave I2S has to prepare the first data to be sent before it receives the clock signal generated by the master. A WS signal of '1' means that the left channel is

sent first. The time to set the I2SE bit to '1' should be at least 2 PCLK clock cycles before the master I2S clock signal on the CK pin.

When the first data is sent, the half-word data is transferred in parallel through the I2S internal bus to the 16-bit shift register, and then the other bits are sent from pin MOSI/SD in order of higher first.

Each time data is transferred from the transmit buffer to the shift register, the flag bit TXE is set to '1' and an interrupt is generated if the TXEIE bit in register SPI_CR2 is '1'.

To ensure continuous audio data transfer, it is recommended to write the next data to be transferred to register SPI_DR before the current transfer is completed. If the new data is still not written to register SPI_DR before the first clock edge representing the next data transfer arrives, the underflow flag bit is set to '1' and an interrupt may be generated; it indicates a software send data error. If the ERRIE bit of register SPI_CR2 is '1', an interrupt is generated when flag bit UDR of register SPI_SR is high.

- Receiving Process

Regardless of the data and channel length, audio data is always received in 16-bit packets, i.e., each time the receive buffer is filled, the flag bit RXNE is set to '1', and an interrupt is generated if the RXNEIE bit in register SPI_CR2 is '1'. Depending on the data and channel length settings, receiving left or right channel data will require one or two transfers of data to the receive buffer. The CHSIDE is updated each time data is received (to be read from SPI_DR), which corresponds to the WS signal generated by the I2S unit. Reading the SPI_DR register will clear the RXNE bit.

When new data is received before the previous received data is read, an overflow is generated and the flag bit OVR is set to '1'; if the ERRIE bit of register SPI_CR2 is '1', an interrupt is generated to indicate that an error has occurred .

25.3.5. Logo position

25.3.5.1. Status Flag

There are 3 status flag bits for the user to monitor the status of the I2S bus.

- Busy flag bit (BSY)

  The BSY flag is set and cleared by hardware (writing this bit has no effect) and this flag bit indicates the status of the I2S communication layer. A '1' in this bit indicates that I2S communication is in progress, with one exception: in main receive mode (I2SCFG=11), the BSY flag is always low during receive. The BSY flag can be used to detect the end of a transmission before the software wants to shut down the SPI module, so that the last transmission is not corrupted, and therefore the following procedure needs to be strictly followed. When a transmission starts, the BSY flag is set to 'will be' unless the I2S module is in master receive mode.

  The flag bit is cleared in the following cases:

  - When the transmission is finished (except for the main transmit mode, in which communication is continuous);
  - When the I2S module is turned off.
  - When communication is continuous:

    In master transmit mode, the BSY flag is always high during the entire transmission;

    In slave mode, the BSY flag goes low for 1 I2S clock cycle between each data item transmission.

- Send buffer empty flag bit (TXE)

  This flag bit is '1' to indicate that the send buffer is empty and new data to be sent can be written to the send buffer. The flag bit is cleared to '0' when there is already data in the transmit buffer. When I2S is turned off (I2SE bit is '0'), this flag bit is also '0'.

- Receive buffer non-empty flag bit (RXNE)

  This flag location '1' indicates that there is valid data received in the receive buffer. This bit is cleared to '0' when the SPI_DR register is read.

- Sound channel flag bit (CHSIDE)

  In transmit mode, this flag bit is refreshed when TXE is high, indicating the channel on which the data sent from the SD pin is located. If an underflow error occurs in the slave transmit mode, the value of this flag bit is invalid and the I2S needs to be turned off and on again before communication can be restarted. In receive mode, this flag bit is refreshed when data is received in register SPI_DR, indicating the channel where the received data is located. If an error occurs (such as an overflow OVR), this flag bit is meaningless and I2S needs to be turned off and then on again (also, if necessary, modify the I2S configuration).

  Under the PCM standard, this flag bit is meaningless in either short or long frame format.

  If the flag bit OVR or UDR of register SPI_SR is '1' and the ERRIE bit of register SPI_CR2 is '1', an interrupt will be generated and the interrupt flag can be cleared later by reading register SPI_SR.

25.3.5.2. Error Flag

The I2S unit has 2 error flag bits.

- Underflow flag bit (UDR)

In slave transmit mode, this flag bit is set to '1' if the new data is still not written to the SPI_DR register when the first clock edge of the data transfer arrives. This flag bit is valid after the I2SMOD position '1' of register SPI_I2SCFGR. If the ERRIE bit of register SPI_CR2 is '1', an interrupt will be generated. This flag bit is cleared by a read operation on register SPI_SR.

- Overflow flag bit (OVR)

If new data is received when the previous received data has not been read out, that is, an overflow is generated, the flag position '1', and if the ERRIE bit of register SPI_CR2 is '1', an interrupt is generated to indicate that an error has occurred. At this time, the contents of the receive cache, will not be refreshed to the new data sent from the transmitting device. A read operation of register SPI_DR returns the last correctly received data. All other 16-bit data sent by the transmitting device after the overflow occurs is lost. This flag bit is cleared by reading register SPI_SR and then register SPI_DR.

25.3.6. Interruption

Table 25-2 I2S interrupt request

Interrupt event	Event Flag Bit	Enable flag bit
Send buffer empty flag bit	TXE	TXEIE
Receive buffer non-empty flag bit	RXNE	RXNEIE
Underflow flag bit	OVR	ERRIE
Overflow flag bit	UDR	

25.4. Register Description

SPI1 register base address: 0x4001 3000

SPI2 register base address: 0x4000 3800

SPI3 register base address: 0x4000 3C00

25.4.1. SPI_CR1 register

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDIMOD	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	BIDIMODE	RW	0	Bidirectional data mode enable 0: select "bi-directional" mode; 1: Select "single line bidirectional" mode. Note: Not used in I2S mode.
14	BIDIOE	RW	0	Output enable in bidirectional mode Together with the BIDIMODE bit, determines the direction of data output in "single line bidirectional" mode 0: output disable (receive-only mode); 1: output enable (transmit-only mode). This "single line" data line is the MOSI pin on the master side and the MISO pin on the slave side. Note: Not used in I2S mode.
13	CRCEN	RW	0	Hardware CRC calculation enable 0: disable CRC calculation; 1: Enable CRC calculation. Note: This bit can be written only when SPI is disabled (SPE=0), otherwise an error occurs. This bit can only be used in full duplex mode. Note: Not used in I2S mode.
12	CRCNEXT	RW	0	Transmit CRC next 0: The next transmit value is from the transmit buffer. 1: The next transmit value is from the transmit CRC register. Note: This bit should be set immediately after the last data is written to the SPI_DR register. Note: Not used in I2S mode.
11	DFF	RW	0	Data frame format 0: transmit/receive using 8-bit data frame format;

Bit	Name	R/W	Reset Value	Function
				1: Use 16-bit data frame format for transmitting/receiving. Note: This bit can be written only when SPI is disabled (SPE=0), otherwise there is an error. Note: Not used in I2S mode.
10	RXONLY	RW	0	Receive only This bit, along with the BIDIMODE bit, determines the direction of transmission in "two-wire bi-directional" mode. In a multiple slave configuration, this bit is set to 1 on the slave device that is not being accessed, so that only the accessed slave device has output, thus not causing data conflicts on the data lines. 0: full duplex (transmit and receive); 1: output disabled (receive only mode). Note: Not used in I2S mode.
9	SSM	RW	0	Software slave management When SSM is set, the level on the NSS pin is determined by the value of the SSI bit. 0: Software slave management is disabled; 1: Software slave management is enabled. Note: Not used in I2S mode.
8	SSI	RW	0	Internal slave select This bit has meaning only when the SSM bit is '1'. It determines the level on the NSS, and I/O operations on the NSS pin are invalid. Note: Not used in I2S mode.
7	LSBFIRST	RW	0	Frame format 0: MSB is sent first; 1: LSB is sent first. Note: The value of this bit cannot be changed while communication is in progress, and an error will occur if the software modifies the communication while it is in progress. Note: Not used in I2S mode.
6	SPE	RW	0	SPI enable 0: disables the SPI device; 1: Enables the SPI device. Note: Not used in I2S mode.
5:3	BR[2:0]	RW	0	Baud rate control 000: fPCLK/2 001: fPCLK/4 010: fPCLK/8 011: fPCLK/16 100: fPCLK/32 101: fPCLK/64 110: fPCLK/128 111: fPCLK/256 Note: The value of this bit cannot be changed while communication is in progress, and an error will occur if the software modifies the communication while it is in progress.

Bit	Name	R/W	Reset Value	Function
				Note: Not used in I2S mode.
2	MSTR	RW	0	Master selection 0: configured as a slave device; 1: Configured as a master device. Note: This bit cannot be modified while communication is in progress. Note: Not used in I2S mode.
1	CPOL	RW	0	Clock polarity 0: SCK remains low in the idle state; 1: SCK remains high in the idle state; 1: SCK is held high in the idle state. Note: The value of this bit cannot be changed while communication is in progress, and an error will occur if the software modifies the communication while it is in progress. Note: Not used in I2S mode.
0	CPHA	RW	0	Clock phase 0: Data is sampled from the first clock edge; 1: Data sampling starts from the second clock edge. Note: The value of this bit cannot be changed while communication is in progress, and an error will occur if the software modifies the communication while it is in progress. Note: Not used in I2S mode.

25.4.2. SPI_CR2 register

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLVFM	LDMA_TX	LDMA_RX	FRXTH	RES				TXEIE	RXNEIE	ERRIE	CLRTXFIFO	RES	SSOE	TXDMAEN	RXDMAEN
RW	RW	RW	RW					RW	RW	RW	W		RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	SLVFM	RW	0	Slave fast mode enable 0: slave normal mode, slave mode supports the fastest SPI clock speed less than pclk/4 1: slave fast mode, can support slave mode SPI clock speed can be up to pclk/4 Note: When the speed of SPI clock is less than pclk/4, this register bit must not be set. Note: Not used in I2S mode.
14	LDMA_TX	RW	0	Last DMA transfer (Send)

Bit	Name	R/W	Reset Value	Function
				This bit is used to define whether the total number of DMA sends is an even or odd number. This function is only valid when data length=8, the TXDMAEN bit is set, and 16-bit write access is available. This bit needs to be written when SPE=0. 0: The total amount of data to be sent is even 1: The total amount of data to be sent is an odd number Note: Not used in I2S mode.
13	LDMA_RX	RW	0	Last DMA transfer (Receive) This bit is used to define whether the total number of DMA receives is an even or odd number. This function is only valid when data length=8, the RXDMAEN bit is set, and 16-bit write access is available. This bit needs SPE=0 to be written. 0: The total amount of data to be received is even 1: The total amount of data to be received is an odd number Note: Not used in I2S mode.
12	FRXTH	RW	0	FIFO Receive Threshold This bit is used to set the RXFIFO threshold value that triggers the RXNE event. 0: If the FIFO level is greater than or equal to 1/2 (16-bit), an RXNE is generated 1: If the FIFO level is greater than or equal to 1/4 (8-bit), the RXNE is generated Note: Not used in I2S mode.
11:8	Reserved	RES	-	Reserved
7	TXEIE	RW	0	Tx buffer empty interrupt enable 0: TXE interrupt is disabled; 1: Allow TXE interrupt, generate interrupt request when TXE flag is set to '1'.
6	RXNEIE	RW	0	RX buffer not empty interrupt enable 0: disable RXNE interrupt; 1: Allow RXNE interrupt, generate interrupt request when RXNE flag is set.
5	ERRIE	RW	0	Error interrupt enable When an error (CRCERR, OVR, MODF) is generated, this bit controls whether to generate an interrupt 0: disable error interrupt; 1: Allow error interrupt.
4	CLRTXFIFO	W		Clear FIFO 0: No effect on FIFO 1: Clear FIFO Software writes 1, hardware automatically clears 0

Bit	Name	R/W	Reset Value	Function
				Write this bit is invalid after spi enable
3	Reserved	RES	-	Reserved
2	SSOE	RW	0	SS output enable 0: Disable SS output in master mode, the device can work in multi-master mode; 1: When the device is on, SS output in master mode is enabled, and the device cannot work in multi-master device mode. Note: Not used in I2S mode.
1	TXDMAEN	RW	0	Tx buffer DMA enable When this bit is set, the TXE flag will issue a DMA request once it is set 0: disables transmit buffer DMA; 1: Enable transmit buffer DMA.
0	RXDMAEN	RW	0	Rx buffer DMA enable When this bit is set, the RXNE flag will issue a DMA request once it is set 0: disables receive buffer DMA; 1: Start receive buffer DMA.

25.4.3. SPI_SR register

Address offset:0x08

Reset value:0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES.			FTLVL		FRLVL		RES	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
			R		R			R	R	R	RCW0	R	R	R	R

Bit	Name	R/W	Reset Value	Function
15:13	Reserved	RES	-	Reserved
12:11	FTLVL	R	0	FIFO transmission level These bits are set and cleared by hardware 00: FIFO empty 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO full (considered full when FIFO threshold is greater than 1/2) Note: This bit cannot be used in I2S mode
10:9	FRLVL	R	0	FIFO reception level These bits are set and cleared by hardware 00: FIFO empty 01: 1/4 FIFO

Bit	Name	R/W	Reset Value	Function
				10: 1/2 FIFO 11: FIFO full Note: This bit cannot be used in I2S mode
8	Reserved	RES	-	Reserved
7	BSY	R	0	Busy flag 0: SPI is not busy; 1: SPI is busy with communication or the transmit buffer is not empty. This bit is set or reset by hardware.
6	OVR	R	0	Overrun flag 0: No overrun error; 1: An overflow error has occurred. This bit is set by hardware and reset by software sequence.
5	MODF	RW	0	Mode fault 0: no mode fault; 1: A mode error occurs. This bit is set by hardware and reset by software sequence. Note: Not used in I2S mode.
4	CRCERR	RCW0	0	CRC error flag 0: The received CRC value matches the value in the SPI_RXCRCR register; 1: The received CRC value and the value in the SPI_RXCRCR register do not match. This bit is set by hardware and reset by software writing '0'. Note: Not used in I2S mode.
3	UDR	R	0	Underrun flag 0: Underrun did not occur; 1: Underrun has occurred. This flag bit is set to '1' by hardware and cleared to '0' by a software sequence. Note: Not used in SPI mode.
2	CHSIDE	R	0	Channel side 0: the left channel needs to be transmitted or received; 1: Need to transmit or receive the right channel. Note: Not used in SPI mode. Meaningless in PCM mode.
1	TXE	R	1	Transmit buffer empty 0: the transmit buffer is non-empty; 1: The transmit buffer is empty.
0	RXNE	R	0	Receive buffer not empty

Bit	Name	R/W	Reset Value	Function
				0: Receive buffer is empty; 1: Receive buffer is not empty.

25.4.4. SPI_DR register

Address offset:0x0C

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	DR[15:0]	RW	0	Data register Data to be sent or already received The data register corresponds to two buffers: one for writing (send buffer) and the other for reading (receive buffer). The write operation writes data to the transmit buffer; the read operation returns the data in the receive buffer. Note on the SPI mode: Depending on the choice of data frame format by the DFF bit of SPI_CR1, data can be sent and received in either 8-bit or 16-bit. To ensure proper operation, the data frame format needs to be determined before SPI is enabled. For 8-bit data, the buffer is 8-bit and only SPI_DR[7:0] will be used when sending and receiving. On receive, SPI_DR[15:8] is forced to 0. For 16-bit data, the buffer is 16-bit and the entire data register, SPI_DR[15:0], is used when sending and receiving.

25.4.5. SPI_CRCPR register

Address offset:0x10

Reset value:0x0007

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

Bit	Name	R/W	Reset Value	Function
15:0	CRCPOLY[15:0]	RW	0	CRC polynomial register This register contains the polynomial used in CRC calculation. Its reset value is 0x0007, other values can be set according to the application. Note: Not used in I2S mode.

25.4.6. SPI_RXCRCR register

Address offset:0x14

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	RxCRC[15:0]	R	0	Receive CRC Register When CRC calculation is enabled, RXCRC[15:0] contains the CRC value calculated based on the received byte. This register is reset when a '1' is written to the CRCEN bit of SPI_CR1. The CRC calculation uses the polynomial in SPI_CRCPR. When the data frame format is set to 8 bits, only the lower 8 bits are involved in the calculation and follow the CRC8 method; when the data frame format is 16 bits, all 16 bits in the register are involved in the calculation and follow the CRC16 standard. Note: Reading this register when the BSY flag is '1' will likely read incorrect values. Note: Not used in I2S mode.

25.4.7. SPI_TXCRCR register

Address offset:0x18

Reset value:0x0000

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

Bit	Name	R/W	Reset Value	Function
15:0	TxCRC[15:0]	R	0	Send CRC register When CRC calculation is enabled, TXCRC[15:0] contains the CRC value calculated based on the byte to be sent. This register is reset when a '1' is written to the CRCEN bit in SPI_CR1. The CRC is calculated using the polynomial in SPI_CRCPR. When the data frame format is set to 8 bits, only the lower 8 bits are involved in the calculation and follow the CRC8 method; when the data frame format is 16 bits, all 16 bits in the register are involved in the calculation and follow the CRC16 standard. Note: Reading this register when the BSY flag is '1' will likely read incorrect values. Note: Not used in I2S mode.

25.4.8. SPI_I2S_CFGR register

Address offset:0x1c

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES.				I2SMOD	I2SE	I2SCFG		PCMSYNC	RES.	I2SSTD		CKPOL	DATLEN		CHLEN
				RW	RW	RW		RW	RW	RW		RW	RW		RW

Bit	Name	R/W	Reset Value	Function
15:12	Reserved	RES	-	Reserved
11	I2SMOD	RW	0	I2S mode selection 0: Selects the SPI mode; 1: Selects I2S mode. Note: This bit can only be set when SPI or I2S is turned off.
10	I2SE	RW	0	I2S enable 0: I2S is disabled; 1: I2S enable. Note: Not used in SPI mode.
9:8	I2SCFG	RW	0	I2S configuration mode 00: transmit from the device; 01: Received by the slave device; 10: Master device transmits; 11: Master device receive. Note: This bit can only be set when I2S is turned off. It is not used in SPI mode.
7	PCMSYNC	RW	0	PCM frame synchronization 0: short frame synchronization; 1: long frame synchronization. Note: This bit is only relevant when I2SSTD = 11 (using PCM standard). It is not used in SPI mode.
6	Reserved	RES	-	Reserved
5:4	I2SSTD	RW	0	I2S standard selection 00: I2S Philips standard; 01: High byte alignment standard (left-aligned); 10: Low byte alignment standard (right-aligned); 11: PCM standard. Note: For correct operation, this bit can only be set when I2S is turned off. It is not used in SPI mode.
3	CKPOL	RW	0	Steady state clock polarity 0: I2S clock is low in the quiescent state; 1: I2S clock stationary state is high.

Bit	Name	R/W	Reset Value	Function
				Note: For correct operation, this bit can only be set when I2S is turned off. It is not used in SPI mode.
2:1	DATLEN	RW	0	Data length to be transferred 00: 16-bit data length; 01: 24-bit data length; 10: 32-bit data length; 11: Not allowed. Note: For correct operation, this bit can only be set when I2S is turned off. It is not used in SPI mode.
0	CHLEN	RW	0	Channel length (number of bits per audio channel) 0: 16 bits wide; 1: 32 bits wide. Writes to this bit are only meaningful if DATLEN = 00, otherwise the channel length is fixed by hardware to 32 bits. Note: For correct operation, this bit can only be set when I2S is turned off. It is not used in SPI mode.

25.4.9. SPI_I2SPR register

Address offset:0x20

Reset value:0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES.						MCKOE	ODD	I2SDIV							
						RW	RW	RW							

Bit	Name	R/W	Reset Value	Function
15:10	Reserved	RES	0	Reserved
9	MCKOE	RW	0	Master clock output enable 0: disables the master clock output; 1: Master clock output enable. Note: For correct operation, this bit can only be set when I2S is turned off. This bit is used only in I2S master device mode. It is not used in SPI mode.
8	ODD	RW	0	Odd factor for the prescaler 0: Actual crossover factor = I2SDIV * 2; 1: Actual crossover factor = (I2SDIV * 2) + 1. Note: For correct operation, this bit can only be set when I2S is turned off. This bit is used only in I2S master device mode.

Bit	Name	R/W	Reset Value	Function
				Not used in SPI mode.
7:0	I2SDIV	RW	0	I2S linear prescaler Disable setting I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 Note: For correct operation, this bit can only be set when I2S is turned off. This bit is used only in I2S master device mode. It is not used in SPI mode.

25.4.10. SPI register map

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CR2	SLVFM	LDMA_TX	LDMA_RX	FRXTH	Res.	Res.	Res.	Res.	TXEIE	RXNEIE	ERRIE	CLRTXFIFO	Res.	SSOE	TXDMAEN	RXDMAEN
	Reset value	0	0	0	0	0				0	0	0	0		0	0	0
0x08	SPI_SR	Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		Res.	BSY	OVR	MODEF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value				0	0	0	0		0	0	0	0	0	0	1	0
0x0C	SPI_DR	DR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPI_CRCPR	CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SPI_RXCRCR	RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPI_TXCRCR	TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	SPI_I2S_CFGR	Res.	Res.	Res.	Res.	I2SMOD	2SE	I2SCFG[1:0]		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN[1:0]		CHLEN
	Reset value					0	0	0	0	0		0	0	0	0	0	0
0x20	SPI_I2SPR	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV							
	Reset value							0	0	0	0	0	0	0	0	0	0

26. Inter-integrated circuit (I2C) interface

26.1. Introduction

The I2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I2C bus. It provides multimaster capability, and controls all I2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm) and Fast-mode (Fm).

DMA can be used to reduce CPU overload.

26.1.1. Main features

- Parallel Bus/I2C Bus Protocol Converter
- Multi-master function: the same interface can be both master and slave device
- I2C Master Device Function
  - Generating the clock
  - Generating start and stop signals
- I2C Slave Device Function
  - Programmable I2C address detection
  - Dual address capability in response to 2 slave addresses
  - Stop bit detection
- Generates and detects 7-bit/10-bit addresses and broadcast calls
- Supports different communication speeds
  - Standard speed (up to 100 kHz)
  - Fast (up to 400 kHz)
- Status Flag:
  - Transmitter/receiver mode flags
  - End of byte send flag
  - I2C bus busy flag
- Error Flag
  - Arbitration loss during master mode
  - Answer (ACK) error after address/data transfer
  - Detected start and stop misalignment
  - Overload and underload (need to disable stretched clock function first)
- 2 interruption vectors
  - 1 event interrupt for successful address/data communication
  - 1 error interrupt
- Optional elongated clock function
- DMA with single-byte buffer
- Configurable PEC (packet error detection) generation or checksum:

- PEC value can be transmitted as the last byte in transmit mode
- PEC error check for the last received byte
- Support SMBus
- 25 ms clock low timeout delay
- 10 ms Cumulative master clock low extension time
- 25 ms Cumulative clock low scaling time for slave devices
- Hardware PEC generation/checking with ACK control
- Address Resolution Protocol (ARP) support

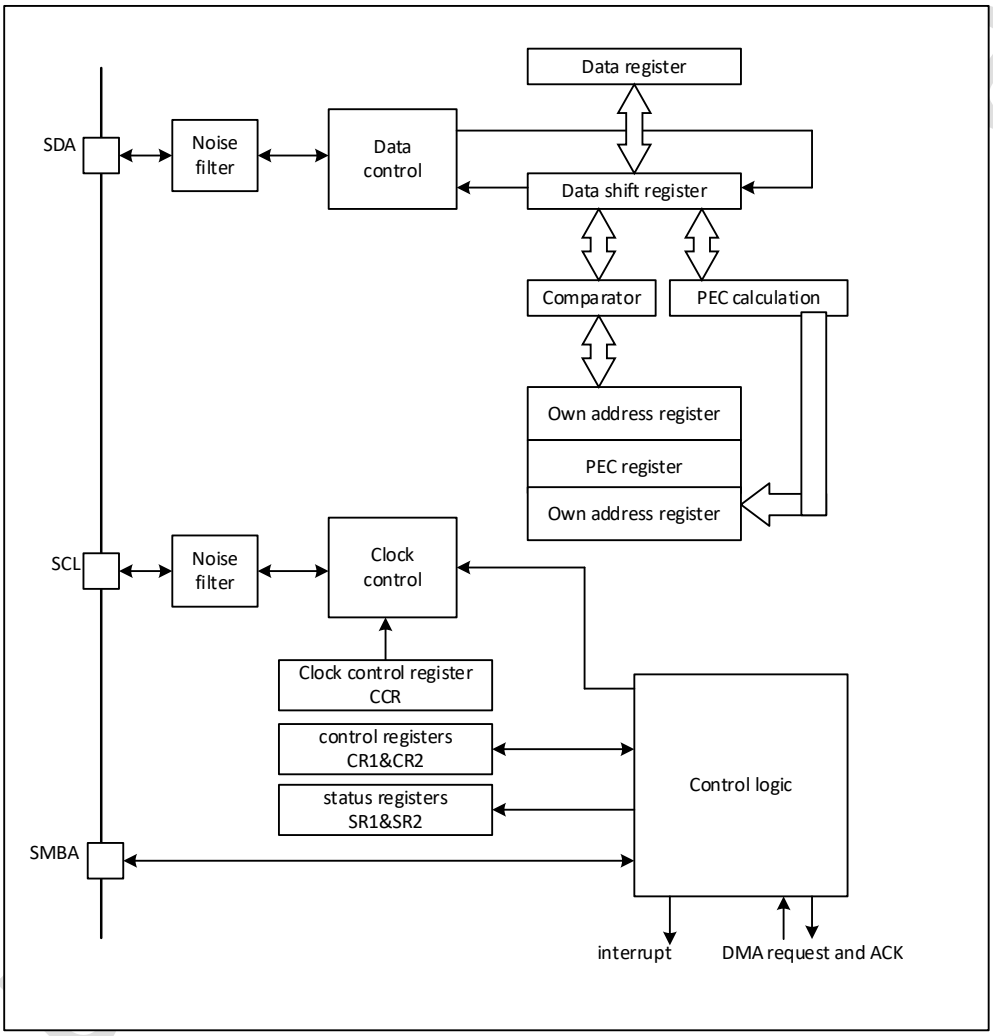


Figure 26-1 I2C module block diagram

26.2. I2C Function Description

(Detailed description by internal functional module. Including but not limited to bus timing, internal modules and their connection relationships, data paths, interrupts, DMA, key state machines, clock reset schemes, etc.)

26.2.1. Introduction

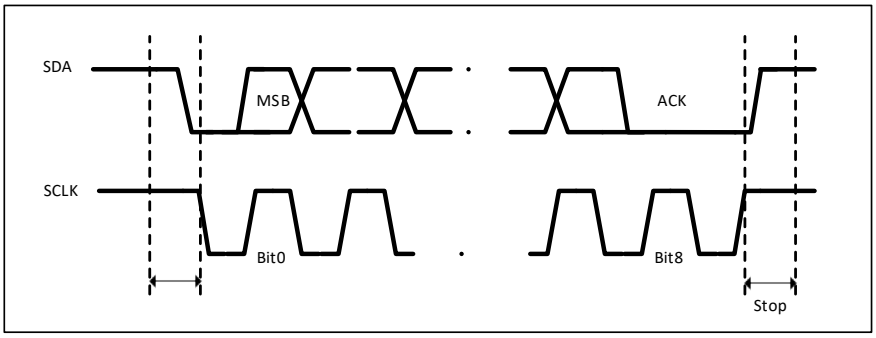


Figure 26-2 I2C Bus Protocol

I2C supports the following four modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

Operates in slave mode by default. The interface automatically switches from slave to master mode after generating a start condition; in the case of allowing multi-master functionality, it switches from master to slave mode when arbitration is lost or a stop signal is generated.

When acting as master mode, the I2C interface initiates data transfer and generates a clock signal. The serial data transfer always starts with a start condition and ends with a stop condition. Both the start condition and the stop condition are generated in the master mode mode by software control.

As slave mode, the I2C interface is able to recognize its own address (7 or 10 bits) and the broadcast call address. The software can control to enable or disable the recognition of the broadcast call address.

Data and address are transmitted in 8 bits (bytes), with the high bit coming first. The 1 or 2 bytes that follow the start condition are the address (1 byte for 7-bit mode and 2 bytes for 10-bit mode). The address is only sent in the main mode mode.

During the 9th clock after 8 clocks of a byte transmission, the receiver must send back an answer bit (ACK) to the sender.

26.2.2. Packet Error Checking (PEC)

The Packet Error Check (PEC) calculator is used to improve the reliability of communication. This calculator uses a programmable polynomial for each bit of serial data.

- PEC calculation is activated by the ENPEC bit of the I2C_CR1 register. pec uses CRC-8 algorithm for all information bytes, including address and read/write bits.
  - On transmit: Set the PEC transmit bit in the I2C_CR1 register on the last TxE event, the PEC will be sent after the current byte.
  - On receive: Set the PEC bit in the I2C_CR1 register on the last RxNE event, if the next received byte is not equal to the internally calculated PEC, the receiver sends a NACK. if it is the master receiver, a NACK will be sent after the PEC, regardless of the result of the proofreading.

- The PECERR error flag/interrupt is available in the I2C_SR1 register.
- If both DMA and PEC calculators are activated:
  - On transmit: When the I2C interface receives an EOT signal from the DMA controller, it automatically sends the PEC after the last byte.
  - On receive: When the I2C interface receives an EOT_1 signal from the DMA, it will automatically take the next byte as PEC and will check it. A DMA request is generated after the PEC is received.
- In order to allow intermediate PEC transfers, there is a control bit (LAST bit) in the I2C_CR2 register to discern if it is indeed the last DMA transfer. If it is indeed the last DMA request from the master receiver, a NACK is automatically sent after the last byte is received.
- The PEC calculation fails when arbitration is lost.

26.2.3. Slave mode

By default, the I2C interface always works in slave mode. To switch from slave mode to master mode, a start condition needs to be generated. In order to generate the correct timing, the input clock for the module must be set in the I2C_CR2 register. The frequency of the input clock must be at least

- Standard mode: 2MHz
- In fast mode: 4MHz

Once the start condition is detected, the address received on the SDA line, is sent to the shift register and compared to the chip's address OAR1 or the general call address (if ENGC=1).

Header segment or address mismatch:

The I2C interface ignores it and waits for another start condition.

Address Match:

The I2C interface generates the following timings:

- If ACK is set '1' by software, an answer pulse is generated
- Hardware sets the ADDR bit and if the ITEVTEN bit is set, an interrupt is generated

In slave mode the TRA bit indicates whether you are currently in receiver mode or transmitter mode

26.2.3.1. Slave transmitter mode

After receiving the address and clearing the ADDR bit, (if the lowest bit of the address byte is a 1) the Slave sends the data (byte) from the DR register, via the internal shift register, to the SDA.

Slave pulls SCL low until the ADDR bit is cleared and the data to be sent has been written to the DR register (refer to EV1, EV3).

When an answer pulse is received: the TxE bit is set by hardware and an interrupt is generated if the ITEVTEN and ITBUFEN bits are set.

If the TxE bit is set but no new data is written to the I2C_DR register before the end of the next data send, the BTF bit is set.Slave pulls SCL low until the BTF bit is cleared by software (after reading I2C_SR1 and then writing to the I2C_DR register).

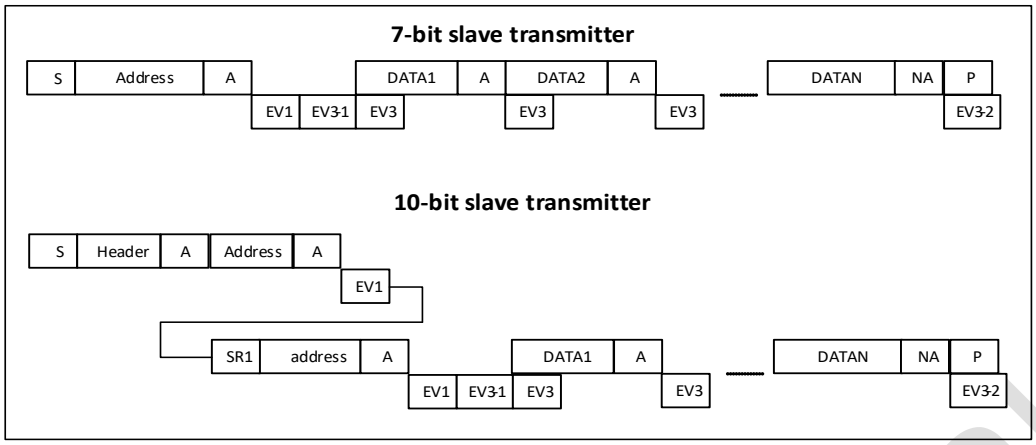


Figure 26-3 Transmission sequence diagram from the transmitter

Description: S=start， Sr=restart， P=stop， A=ack，NA=nack，EVx=event (Interrupt generated when ITEVTEN=1)

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event(with interrupt if ITEVFEN= 1)

EV1: ADDR=1, clear the ADDR bit by reading the SR1 register first, then the SR2 register

EV3-1: TxE=1, shift register empty, data register empty, write Data1 to DR register

EV3: TxE=1, shift register not empty, data register empty, write to DR register (Data2) to clear TxE

EV3-2: AF=1; software writes 0 to AF bit to clear this bit

26.2.3.2. slave receive mode

After receiving the address and clearing ADDR, (if the lowest bit of the address byte is 0) the slave will store the byte received from the SDA line into the DR register via the internal shift register. the I2C interface performs the following operations after each byte is received:

- If the ACK bit is set, an answer pulse is generated
- Hardware sets RxNE=1. If the ITEVTEN and ITBUFEN bits are set, an interrupt is generated.

If RxNE is set and the DR register is not read before the end of receiving new data, the BTF bit is set and the slave pulls SCL low until the BTF is cleared (I2C_SR1 read followed by a read of the I2C_DR register) (see below).

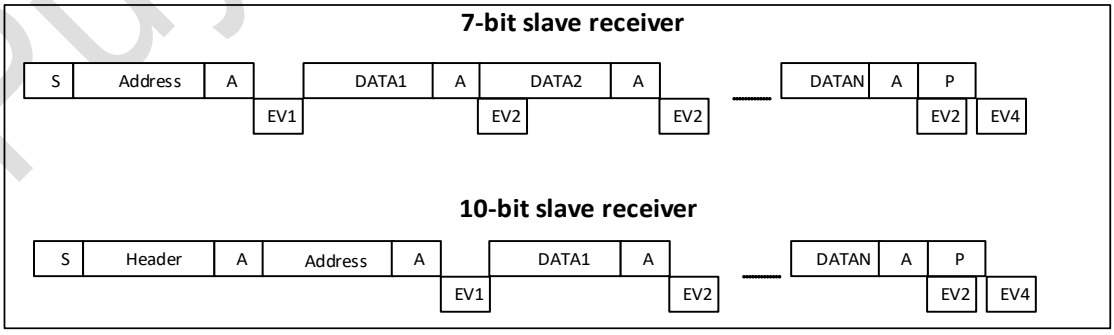


Figure 26-4 Transmission sequence diagram from the receiver

Description: S=start， Sr=restart， P=stop， A=ack，NA=nack，EVx=event (Interrupt generated when ITEVTEN=1)

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledged, EVx= Event(with interrupt if ITEVFEN=1)

EV1: ADDR=1, read SR1 first, then read SR2 to zero ADDR

EV2: RxNE=1, read the DR register to clear the bit.

EV4: STOPF=1, clear this bit by reading SR1 register first and then writing CR1 register.

Note:

1) EV1 event pulls down SCL until the end of the corresponding software sequence.
2) The EV2 software sequence must be completed before the current byte transfer is complete.
3) After the user checks the contents of the SR1 registers, a complete clear sequence should be performed for each flag bit found to be set. e.g., the ADDR and STOPF flag bits require the following sequences:

   If ADDR=1, read SR first, then SR2; if STOPF=1, read SR1 first, then write CR1.

   This is done to ensure that if both ADDR and STOPF are found to be set, both are cleared.

26.2.3.3. Shutting down communication

After the last data byte is transmitted, the master generates a stop condition, which is detected by the slave:

- Hardware sets STOPF and generates an interrupt if the ITEVTEN bit is set.
- Clearing the STOPF bit is accomplished by reading SR1 first and then writing CR1.

26.2.4. Master Mode

In Master mode, the I2C interface initiates the data transfer and generates the clock signal. Serial data transfers always start with a start condition and end with a stop condition. When a start condition is generated on the bus via the START bit, the device enters master mode.

The following is the sequence of operations required by the MASTER mode:

- Set the module's input clock in the I2C_CR2 register to generate the correct timing
- Configure the clock control registers
- Configure the rise time register
- Program the I2C_CR1 register to start the peripherals
- Set the START bit in the I2C_CR1 register to 1 to generate the start condition

The input clock frequency of the I2C module must be at least:

- In standard mode: 2MHz
- In fast mode: 4MHz

26.2.4.1. Host generates clock

The CCR register generates the high and low levels of SCL with either a rising or falling edge. Since the slave may stretch the SCL signal, after the SCL rising edge is generated, the master checks the SCL signal from the bus when the time programmed in the TRISE register arrives.

- If SCL is low, it means that the slave is stretching the SCL bus and the high counter stops counting until SCL is detected high. This is to ensure a minimum high time for the SCL parameter.

- If SCL is high, the high counter keeps counting.

In fact, even if the slave doesn't stretch the SCL, such a feedback loop takes some time from the time the SCL rising edge is generated, to the time the SCL rising edge is detected. The time of this loop is related to the rise time of the SCL (VIH data detection of the SCL), plus the analog noise filtering of the SCL input path, and the internal SCL synchronization of the chip due to the clocking with the APB. The maximum time of the feedback loop is programmed in the TRISE register, so the frequency of the SCL remains stable regardless of the SCL rise time.

26.2.4.2. commencement condition

When BUSY=0, set START=1, the I2C interface will generate a Start condition and switch to master mode (MSL is set).

Note:

1) Setting the START bit in master mode will generate a ReStart condition by hardware after the current byte has been transferred.
2) Once the Start condition is issued: the SB bit is set by hardware and an interrupt is generated if the ITEVTEN bit is set. master reads the SR1 register and then writes the slave address to the DR register.

26.2.4.3. Slave Address Transmission

The slave address is sent to the SDA line via an internal shift register.

- When in 10-bit address mode, sending a header segment sequence generates the following events:
  - The ADD10 bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
  - The master device then waits for a read of the SR1 register, followed by a write of the second address byte to the DR register.
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
  - The master device then waits once to read the SR1 register, followed by the SR2 register.
- In 7-bit address mode, an address byte is sent.
  - Once this address byte has been sent, the ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
  - The master device then waits for a read of the SR1 register, followed by a read of the SR2 register.
  - Depending on the LSB bit sent from the slave address, the master device decides whether to enter transmitter mode or receiver mode.
- In 7-bit address mode.
  - To enter transmitter mode, the master device sends the slave address with LSB equal to 0.
  - To enter receiver mode, the master device sends the slave address with LSB equal to 1.
- In 10-bit address mode
  - To enter transmitter mode, the master device sends the header byte (11110xx0) followed by the slave address with the LSB bit equal to zero. (The xx in the header byte is the highest 2 bits of the 10-bit address.)

- To enter receiver mode, the master sends the header byte (11110xx0) followed by the slave address with the LSB bit equal to 0. The master sends the header byte (11110xx0) followed by the slave address with the LSB bit equal to 0. It then retransmits a start condition followed by the header byte (11110xx1).

(xx in the header byte is the highest 2 bits of the 10-bit address) The TRA bit indicates whether the master device is in receiver or transmitter mode

26.2.4.4. Master transmitter

After sending the address and clearing the ADDR bit, the master device master sends bytes from the DR register to the SDA line via the internal shift register. master waits until the first data byte is written to the DR register.

When an ACK pulse is received, the TxE bit is set by hardware and an interrupt is generated if the INEVFEN and ITBUFEN bits are set.

If TxE is set and no new data byte is written to the DR register before the end of the last data transmission, BTF is set by hardware. The I2C interface will hold SCL low until BTF is cleared (read of I2C_SR1 followed by a write of the I2C_DR register).

26.2.4.5. Shutting down communications

After writing the last byte in the DR register, a stop condition is generated by setting the STOP bit, and then the I2C interface will automatically return to slave mode (MSL bit clear).

Note:

1) When the TxE or BTF position bits are present, the stop condition should be scheduled for the occurrence of an EV8_2 event.

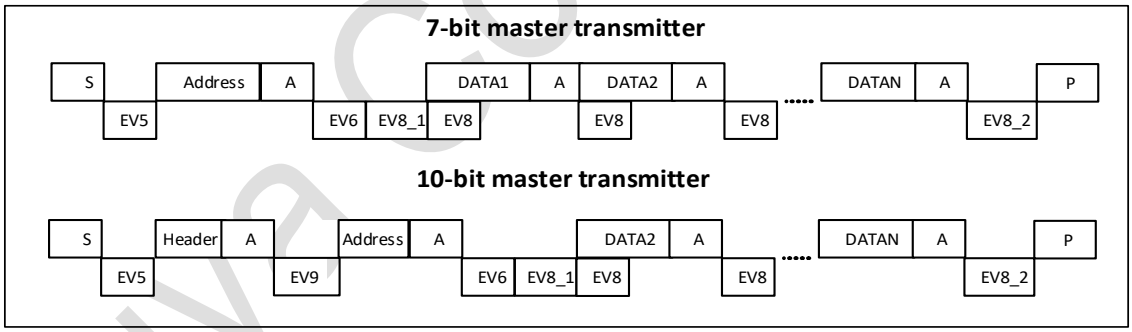


Figure 26-5 Transmission sequence diagram of the main transmitter

Description: S=start， Sr=restart， P=stop， A=ack，NA=nack，EVx=event (Interrupt generated when ITEVTEN=1)

EV5: SB=1, reading SR1 and then writing the address to the DR register will clear this event

EV6: ADDR=1, read SR1 and then read SR2 to clear this event

EV8_1: TXE=1, shift register empty

EV8: TXE=1, write DR register clears this event

EV8_2: TXE=1, BTF=1, cleared by hardware when stop condition is generated

EV9: ADDR10=1, read SR1 then write DR register to clear the event

Note:

1) EV5, EV6, EV8_1 and EV8_2 events stretch SCL low until the execution of the corresponding software sequence is complete.
2) The EV8 software sequence must be executed before the current byte transmission is complete. In the EV8 software sequence cannot be completed before the end of the currently transmitted byte, it is recommended to use BTF instead of TxE, which has the disadvantage of slowing down the communication.

26.2.4.6. Master receiver

After sending the address and clearing the ADDR, the I2C interface enters the main receiver mode. In this mode, the I2C interface receives data bytes from the SDA line and sends them through the internal shift register to the DR register. After each byte, the I2C interface performs the following operations in sequence:

- If the ACK bit is set, an answer pulse is issued.
- Hardware sets RxNE=1 and an interrupt is generated if the INEVFEN and ITBUFEN bits are set.

If the RxNE bit is set and the data in the DR register is not read before the end of receiving new data, the hardware will set BTF=1 and the I2C interface will hold SCL low until BTF is cleared; reading I2C_SR1 followed by reading the I2C_DR register will clear the BTF bit.

26.2.4.7. Shutting down communication

Method 1: This method is used when I2C is set as the highest priority interrupt in the application.

The Master sends a NACK after receiving the last byte from the Slave, and after receiving the NACK, the Slave releases control of the SCL and SDA lines, and the Master can then send a Stop/Restart condition.

1) In order to generate a NACK pulse after the last byte is received, the ACK bit must be cleared after the penultimate data byte is read (after the penultimate RxNE event).
2) In order to generate a STOP/restart condition, software must set the STOP/START bit after reading the penultimate data byte (after the penultimate RxNE event).
3) When only one byte is received, just after EV6 (after clearing ADDR when EV6_1) the answer and stop condition generation bits have to be turned off. After the stop condition is generated, the I2C interface automatically returns to slave mode (MSL bit is cleared).

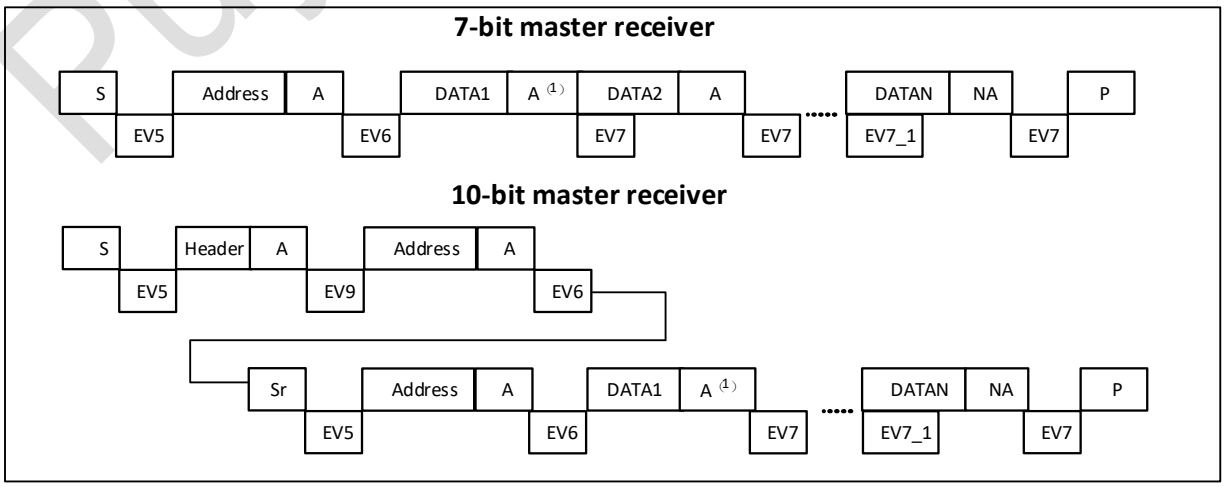


Figure 26-6 Method 1: Timing during main mode reception

Description: S=start， Sr=restart， P=stop， A=ack，NA=nack，EVx=event(Interrupt generated when ITEVTEN=1)

EV5: SB=1, read SR1, then write DR register, this bit is cleared to zero

EV6: ADDR=1, read SR1, then read SR2, this bit is cleared

EV6_1: no associated flag event, used as 1 byte receive only.

EV7: RxNE=1, read DR register, this bit is cleared to zero

EV7_1: RxNE=1, read DR register, write ACK=0 and set to STOP

EV9: ADDR10=1, read SR1 then write DR register to clear this event.

NOTE:

1) If a single byte is received, the place labeled (1) above will be NA
2) EV5, EV6 events, stretch SCL low until the corresponding software sequence is executed.
3) EV7 software sequences must be executed before the current byte is sent. In EV7, the software sequence cannot be managed until the transmission of the currently transmitted byte is complete. It is recommended to use BTF instead of RXNE, which has the disadvantage of slowing down the communication.
4) The softwaresequence in EV6_1 or EV7_1 must be completed before the ACK of the current byte transmission.

Method 2: This method is used if the I2C interrupt is not the highest priority in the application or if a query is used.

With this method, DataN_2 is not read, so the communication is stretched after DataN-1 (both RxNE and BTF are set). The ACK bit is then cleared before reading DataN_2 in the DR register to ensure that it is cleared before DataN ACK. After this, after reading DataN-2, the STOP/START bit is set and DataN-1 is read.After RxNE is set, DataN is read.

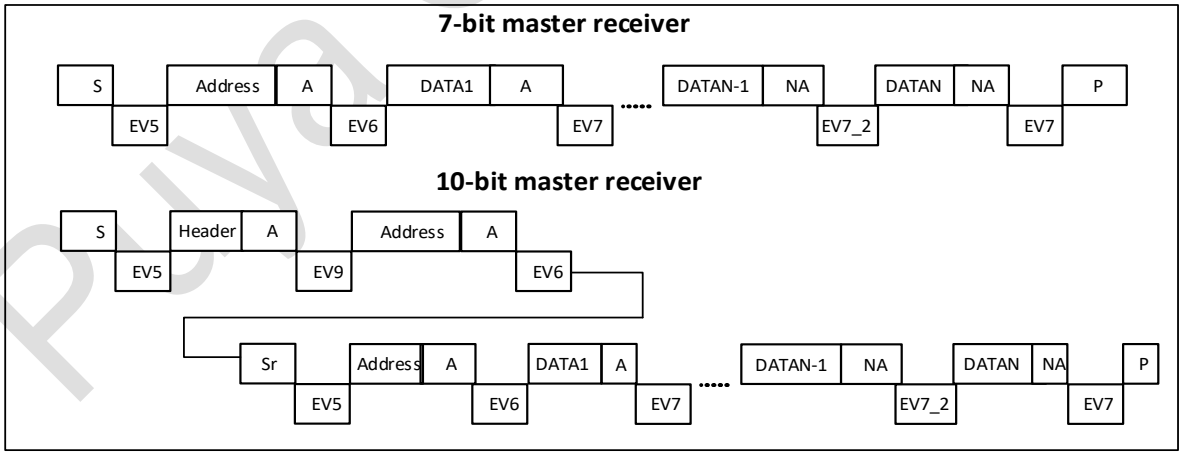


Figure 26-7 Method 2:Timing during main mode reception for N>2

Description: S=start， Sr=restart， P=stop， A=ack，NA=nack，EVx=event (Interrupt generated when ITEVTEN=1)

EV5: SB=1, read SR1 register first, then write DR register, clear this bit

EV6: ADDR, read SR1 first, then read SR2, clear this bit

EV7: RxNE=1, read DR register to clear this bit

EV7_2: BTF=1, DataN-2 exists in DR register, DataN-1 exists in shift register, write ACK=0, read DataN-2 in DR register. set STOP, read DataN-1

EV9: ADDR10=1,read SR1 then write DR register to clear the event

Note:

1) EV5, EV6 events, stretch SCL low until the execution of the corresponding software sequence is completed.

2) EV7 software sequences must be executed before the current byte transmission is complete. In EV7, the software sequence cannot be managed until the transmission of the currently transmitted byte is complete. It is recommended to use BTF instead of RXNE, which produces the disadvantage of slowing down the communication.

- When 3 bytes are to be read away:
  - RxNE = 1 => Nothing (DataN - 2 not read).
  - DataN - 1 received
  - BTF = 1, shift and data registers are both full: DR register holds DataN -2, shift register holds DataN -1 = > SCL pull down: no other data to be received on the bus.
  - Clear the ACK bit
  - Read DataN - 2 in DR register = > This initiates reception of DataN in the shift register.
  - DataN reception complete (with a NACK)
  - Write START or STOP bit
  - Read DataN - 1
  - RxNE = 1
  - Read DataN

The above flow is described for N > 2. 1 byte and 2 byte receipts are handled differently, see the following description:

- The case of 2-byte reception
  - Setting the POS and ACK bits
  - Wait for ADDR to be set
  - Clear ADDR bit
  - Clear ACK bit
  - Wait for BTF to be set
  - Write STOP bit
  - Read DR twice

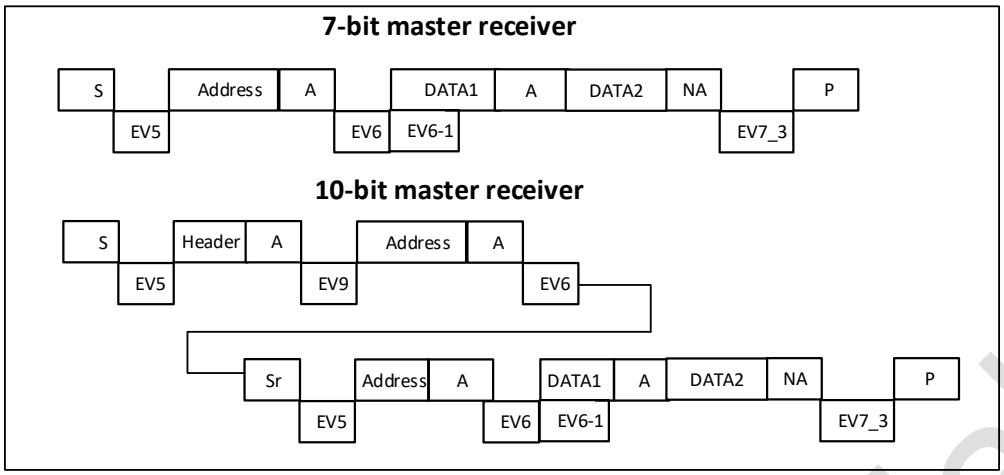


Figure 26-8 Timing during main mode reception with N=2

Description: S=start， Sr=restart， P=stop， A=ack，NA=nack，EVx=event (Interrupt generated when ITEVTEN=1)

EV5: SB=1, read SR1 register first, then write DR register, clear this bit

EV6: ADDR=1, read SR1 register first, then read SR2 register, clear ADDR bit

EV6_1: No associated flag bit event. ACK should be cleared after EV6, i.e. after the address is cleared

EV7_3: BTF=1, write STOP=1, followed by two DR reads (Data1 and Data2)

EV9: ADDR10=1,read SR1 then write DR register to clear the event

Note:

1) EV5, EV6 events, stretch SCL low until the execution of the corresponding software sequence is completed
2) The software sequence for EV6_1 must be completed before the ACK of the current byte transfer

■ Single byte reception

- Clear the ACK bit in the ADDR event.
- Clear ADDR
- Write STOP or START bit
- Read data after the RxNE flag is set

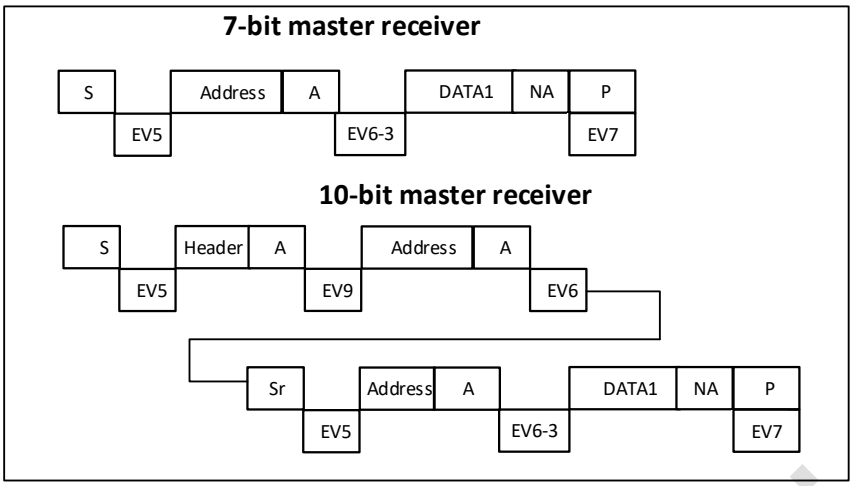


Figure 26-9 Timing during main mode reception when N=1

Description: S=start， Sr=restart， P=stop， A=ack，NA=nack，EVx=event (Interrupt generated when ITEVTEN=1)

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event(with interrupt if ITEVFEN= 1)

EV5: SB=1, read SR1 register first, then write DR register, clear the bit

EV6_3: ADDR=1, write ACK=0. Read SR1 register first, then SR2 register, clear the ADDR bit. After ADDR is cleared, write STOP=1

EV7: RxNE=1, read DR register to clear this bit

EV9: ADDR10=1, read SR1 then write DR register to clear this event

Note:

EV5, EV6, EV8_1, and EV8_2 events, stretch SCL low until the corresponding software sequences are executed.

26.2.5. Error status

26.2.5.1. Bus Error

A bus error occurs when the I2C interface detects an external stop or start condition during an address or data byte transfer. At this time:

- The BERR bit is set to '1'; if the ITERREN bit is set, an interrupt is generated;
- In slave mode: the data is discarded and the hardware releases the bus:
  - In case of an erroneous Start condition, the slave considers it a Restart and waits for an address or stop condition
  - In case of an erroneous Stop condition, the slave operates according to the normal Stop condition while the hardware releases the bus.
- In master mode: the hardware does not release the bus and does not affect the current transmission status. It is up to the software to decide whether or not to abort the current transfer.

26.2.5.2. Answer Fault (AF)

An answer error is generated when the interface detects a no answer bit. At this time:

- The AF bit is set and an interrupt is generated if the ITERREN bit is set!
- When the transmitter receives a NACK, the communication must be reset:
  - If in slave mode, the hardware releases the bus.
  - If in master mode, software must generate a stop condition or REPEATED START.

26.2.5.3. Arbitration lost (ARLO)

An arbitration loss error is generated when the I2C interface detects a loss of arbitration. at this point:

- The ARLO bit is set by hardware and an interrupt is generated if the ITERREN bit is set
- The I2C interface automatically returns to slave mode (MSL bit is cleared). When an I2C interface loses arbitration, it cannot respond to its slave address in the same transfer, but it can respond after the master of the winning bus sends a REPEATED START condition
- Hardware release the bus

26.2.5.4. Overload/Underload Error (OVR)

In slave mode, if clock extension is disabled (nostretch = 1) and the I2C interface is receiving data, an overrun error occurs when it has received a byte (RxNE = 1) but the previous byte of data in the DR register has not yet been read.

At this time:

- The last received data is discarded
- In case of overrun error, the software should clear the RxNE bit and the transmitter should retransmit the last byte sent

In slave mode, an underrun error occurs if the clock extension is disabled and the I2C interface is sending data when the new data has not been written to the DR register (TxE=1) before the clock for the next byte arrives. At this time:

- The previous byte in the DR register will be sent out repeatedly
- The user should determine that the receiving end should discard duplicate received data in the event of an underrun error. The sender should update the DR register at the specified time according to the I2C bus standard

When sending the first byte, it must be written to the DR register after clearing the ADDR and before the first SCL rising edge; if this cannot be done, the receiver should discard the first data.

26.2.6. DMA function

DMA requests (when enabled) are used only for data transfers. A DMA request is generated when the data register becomes empty when transmitting, or when it becomes full when receiving.DMA must be initialized and enabled before the current byte transfer ends.The DMAEN bit (in the I2C_CR2 register) must be enabled before an ADDR event occurs.

In master or slave mode, when clock extension is enabled, the DMAEN bit can be set during an ADDR event prior to clearing ADDR. the DMA request must be responded to before the current byte transfer is complete. When the DMA transfer data length reaches the value set by the DMA, the DMA controller sends an EOT (End of transfer) to the I2C and generates a transfer complete interrupt (if the interrupt enable bit is active):

- Master transmitter: In the EOT interrupt service program, it is necessary to disable the DMA request and then set the stop condition after waiting for the BTF event.
- Master receiver: when the number of data to be received is greater than or equal to 2, the DMA controller sends a hardware signal EOT_1, which corresponds to a DMA transfer (byte count - 1). If the LAST bit is set in the I2C_CR2 register, the hardware will automatically send a NACK for the next byte after EOT_1 is sent. where interrupts are permitted, the user can generate a stop condition in the interrupt service program for the completion of a DMA transfer.

26.2.6.1. Using DMA to send

DMA mode can be enabled by setting the DMAEN bit in the I2C_CR2 register. As soon as the TxE bit is set, data will be loaded into the I2C_DR register by DMA, from the preset memory area. To assign a DMA channel to the I2C, the following steps must be performed (x is the channel number):

1. Set the I2C_DR register address in the DMA_CPARx register. Data will be transferred from memory to this address after each TxE event.
2. Set the memory address in the DMA_CMARx register. Data will be transferred from this memory to I2C_DR after each TxE event.
3. Set the number of bytes to be transferred in the DMA_CNDTRx register. This value is decremented after each TxE event.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register.
5. Set the DIR bit in the DMA_CCRx register and depending on the application requirements it can be configured to issue an interrupt request when the entire transfer is half or all the way complete.
6. Activate the channel by setting the EN bit on the DMA_CCTx register.

When the number of data transfers set in the DMA controller has been completed, the DMA controller sends an end-of-transfer EOT/EOT_1 signal to the I2C interface. A DMA interrupt will be generated if interrupts are allowed.

Note: Do not set the ITBUFEN bit of the I2C_CR2 register if DMA is used for transmission.

26.2.6.2. Receive using DMA

DMA receive mode can be activated by setting the DMAEN bit in the I2C_CR2 register. Each time a data byte is received, the data in the I2C_DR register will be transferred by DMA to the set memory area (refer to the DMA description). To set up the DMA channel for I2C reception, the following steps must be performed (x is the channel number):

1. Set the address of the I2C_DR register in the DMA_CPARx register. Data will be transferred from this address to the memory area after each RxNE event.
2. Set the memory area address in the DMA_CMARx register. Data will be transferred from the I2C_DR register to this storage area after each RxNE event.
3. set the number of bytes to be transferred in the DMA_CNDTRx register. This value will be decremented after each RxNE event.
4. Configure the channel priority with PL[0:1] in the DMA_CCRx register.
5. clear the DIR bit in the DMA_CCRx register, which can be set to issue an interrupt request when half or all of the data transfer is completed, depending on application requirements.

6. Set the EN bit in the DMA_CCRx register to activate the channel.

When the number of data transfers set in the DMA controller has been completed, the DMA controller sends an end-of-transfer EOT/EOT_1 signal to the I2C interface. A DMA interrupt will be generated if interrupts are allowed.

Notes:

1) Do not set the ITBUFEN bit of the I2C_CR2 register if DMA is used for reception.

Table 26-10 I2C Interrupt Request

Interrupt event	Event Flag	Enable control bits
START has been sent (Master)	SB	ITEVTEN
Address has been sent(Master) / Address matched(Slave)	ADDR	
ADD10	10-bit header segment sent (master)	
Stop detection interrupt flag(Slave)	STOPF	
Transfer Complete Reload	BTF	
Receive buffer not empty	RxNE	ITEVTEN 和 ITBUFEN
Transmit buffer empty	TxE	
Bus error	BERR	ITERREN
Arbitration loss(Master)	ARLO	
ACK Failed	AF	
Overrun/Underrun	OVR	
PEC error	PECERR	
Timeout/Tlow Error	TIMEOUT	
SMBus alert	SMBALERT	

26.2.7. System Management Bus SMbus

26.2.7.1. Introduction

The System Management Bus (SMBus) is a two-wire interface. Through it, devices can communicate with each other and with other parts of the system. It is based on the I2C operating principle.SMBus provides a control bus for system and power management related tasks. A system utilizing SMBus can communicate with multiple devices without the use of separate control lines.

The System Management Bus (SMBus) standard involves three types of devices. Slave devices, which receive or respond to commands. Master devices, devices used to issue commands, generate clocks, and terminate transmissions. Host, a specialized master device that provides the primary interface to the system CPU. The host must have master-slave capabilities and must support the SMBus notification protocol. Only one host is allowed in a system.

Using the system management bus, a device can provide manufacturer information, tell the system its model/part number, save the status of a suspend event, report different types of errors, receive control parameters, and return its status.SMBus provides a control bus for system and power

management related tasks. SMBus provides the control bus for system and power management related tasks.

The I2C1 module in this project supports SMbus/PMbus functionality, SMbus versus I2C.

26.2.7.2. Differences between SMBus and I2C

SMBus	I2C
Maximum transmission speed 100kHz	Maximum transmission speed 400kHz
Minimum transmission speed 10kHz	No minimum transmission speed
35ms clock low timeout	No clock timeout
Fixed logic level	Logic level is determined by VDD
Different address types (reserved, dynamic, etc.)	7-bit, 10-bit, and Broadcast Call Slave Address Types
Different bus protocols (quick commands, handling calls, etc.)	No bus protocol

26.2.7.3. Similarities between SMBus and I2C

- 2-wire bus protocol (1 clock, 1 data) + optional SMBus reminder line
- Master-slave communication, master provides clock
- Multi-master functionality
- SMBus data format similar to I2C 7-bit address format

26.2.7.4. Address Resolution Protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning each slave device a new unique address.

ARP has the following attributes:

- Address assignment utilizes the standard SMBus physical layer arbitration mechanism
- The assigned address remains the same while the device maintains power, allowing the device to retain its address in the event of a power failure.
- There is no additional SMBus packing overhead after an address is assigned (that is, it takes the same amount of time to access a device with an assigned address as it does to access a device with a fixed address).
- Any SMBus master device can traverse the bus.

26.2.7.5. SMBus alert mode (ALERT)

The SMBus reminder is an optional signal with an interrupt line for devices that wish to extend their control capabilities at the expense of a pin.SMBALERT is a line-and-signal like the SCL and SDA signals.SMBALERT is typically used in conjunction with the SMBus broadcast call address. SMBus-related messages are 2-byte.

A single slave device can signal the host that it wishes to communicate via SMBALERT, which is accomplished by setting the ALERT bit on the I2C_CR1 register. The host handles the interrupt and accesses all SMBALERT devices via the Alert Response Address ARA (address 0001100x). Only those devices that have pulled SMBALERT low can respond to the ARA. This status is identified by the SMBALERT status flag in the I2C_SR1 register. The host performs a modified receive byte

operation. The 7-bit device address provided by the transmitting device is placed in the seven highest bits of the byte, and the eighth bit can be either 0 or 1.

If more than one device pulls SMBALERT low, the highest-priority device (with the smallest address) wins the right to communicate through standard arbitration during address transfer. This device must not pull its SMBALERT low again after the slave address has been acknowledged. the host knows that it needs to read the ARA again if it still sees SMBALERT low when the message transfer is complete. hosts that do not perform the SMBALERT signaling can periodically access the ARA.

26.2.7.6. Timeout error (TIMEOUT)

There are many differences between I2C and SMBus in terms of timing specifications.SMBus defines a clock low timeout, 35ms timeout.SMBus specifies TLOW:SEXT as the cumulative clock low extension time for the slave.SMBus specifies TLOW:MEXT as the cumulative clock low extension time for the master.SMBus specifies TLOW:MEXT as the cumulative clock low extension time for the master.SMBus specifies TLOW:MEXT as the cumulative clock low extension time for the master. timeout. Refer to the SMBus specification version 2.0 (http://smbus.org/specs/) for more timeout details. The status flag Timeout or Tlow error in I2C_SR1 indicates the status of this feature.

26.2.7.7. PMbus

PMbus is based on SMbus, the transmission logic is exactly the same as SMbus, the difference is that PMbus defines some functions related to power management (done by software).

26.3. Register Description

The registers can be half-word or word accessed.

26.3.1. I2C Control register 1 (I2C_CR1)

Address offset:0x00

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Res.	SMBUS	PE
RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	RW

Bit	Name	R/W	Reset Value	Function
15	SWRST	RW	0	Software Reset. When set, I2C is in a reset state. Make sure the I2C pins are released and the bus is idle before the reset is released. 0: I2C module is not in reset state 1: I2C module is in reset state Note: This bit can be used to re-initialize I2C in the ERROR or LOCKED state. e.g., if the BUSY

Bit	Name	R/W	Reset Value	Function
				bit is 1 and no stop condition is detected on the bus.
14	Reserved	RES	-	Reserved
13	ALERT	RW	0	SMBus Alert. 0: Release the SMBAlert pin to make it high. The alert response address header immediately follows the NACK signal; 1: Drive the SMBAlert pin to make it low. The reminder response address header immediately follows the ACK signal; Cleared by hardware when PE=0.
12	PEC	RW	0	Packet error checking Software can set and clear this bit, and hardware can clear this bit in the following cases: when a PEC is transmitted followed by a start condition, or a stop condition, or when PE=0; 0: No PEC transmission 1: PEC transmission (in transmit or receive mode) Note: The calculation of PEC is disabled when arbitration is lost.
11	POS	RW	0	ACK/PEC position (for data reception), this register can be set/cleared by software or cleared by hardware when PE=0. 0: ACK bit controls the (N)ACK of the byte currently being received in the shift register. the PEC bit indicates that the byte currently in the shift register is PEC. 1: The ACK bit controls the (N)ACK of the next byte being received in the shift register. the PEC bit indicates that the next byte being received in the shift register is a PEC. Note: The POS bit can only be used in a 2-byte receive configuration and must be configured prior to receiving data. In order to NACK the 2nd byte, the ACK bit must be cleared after clearing ADDR. In order to detect PEC for the 2nd byte, the PEC bit must be set on an ADDR stretch event after the POS bit is configured.
10	ACK	RW	0	Answer Enable. This register can be set/cleared by software or cleared by hardware when PE=0.

Bit	Name	R/W	Reset Value	Function
				0: No answer return 1: Returns an answer after a byte is received. (matching address or data)
9	STOP	RW	0	A stop condition is generated and this register can be set/cleared by software or cleared by hardware when a stop condition is detected, or set by hardware when a timeout error is detected. In master mode: 0: No stop condition generated 1: Stop condition generated after the current byte is transmitted or after the current start condition is issued In slave mode: 0: No stop condition generated 1: Release SCL and SDA lines after current byte transmission
8	START	RW	0	The start condition is generated. This register can be set/cleared by software, or cleared by hardware when the start condition is issued or when PE=0. Master Mode: 0: No start condition generation 1: Repeat start condition generation Slave mode: 0: No start condition generated 1: Generate start condition when the bus is idle (and automatically switched to master mode by hardware)
7	NOSTRETCH	RW	0	Inhibit Clock Extension (Slave). When the ADDR or BTF flag is set, this bit is used for slave to disable clock extension until reset by software. 0: Clock extension is allowed 1: Prohibit clock extension
6	ENGC	RW	0	Broadcast call enable. 0: Disable broadcast call. Respond with a NACK to address 00h 1: Broadcast call is allowed. Respond with ACK to address 00h
5	ENPEC	RW	0	PEC Enable.

Bit	Name	R/W	Reset Value	Function
				0: Disable PEC calculation 1: Enable PEC calculation
4	ENARP	RW	0	ARP enable. 0: ARP disabled; 1: Enable ARP; If SMBTYPE=0, use the default address of the SMBus device; If SMBTYPE=1, use the primary address of the SMBus.
3	SMBTYPE	RW	0	SMBus type. 0: SMBus device; 1: SMBus host;
2	Reserved	RES	-	Reserved
1	SMBUS	RW	0	SMBus mode. 0: I2C mode; 1: SMBus mode;
0	PE	RW	0	I2C module enable. 0: Disable 1: I2C enable Depending on the configuration of the SMBus bit, the corresponding I/O port needs to be configured for multiplexing. Note: If communication is in progress when this bit is cleared, the I2C module is disabled and returns to the idle state at the end of the current communication. Since PE=0 at the end of communication, all bits are cleared. In master mode, the bit must not be cleared before the end of communication.

26.3.2. I2C Control register 2 (I2C_CR2)

Address offset:0x04

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Res.	Res.	FREQ[5:0]					
			RW	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:13	Reserved	RES	-	Reserved
12	LAST	RW	0	DMA last transfer.

Bit	Name	R/W	Reset Value	Function
				0: EOT of the next DMA is not the last transmission 1: EOT of the next DMA is the last transmission Note: This bit is used in master receive mode to enable a NACK to be generated on the last data received.
11	DMAEN	RW	0	DMA request enable. 0: Disable DMA request; 1: DMA request is allowed when TxE=1 or RxNE=1.
10	ITBUFEN	RW	0	Buffer interrupt enable. 0: When TxE=1 or RxNE=1, no interrupt is generated 1: When TxE=1 or RxNE=1, event interrupt is generated (regardless of DMAEN value)
9	ITEVTEN	RW	0	Event interrupt enable. 0: Disable 1: Event interrupt enabled This interrupt will be generated under the following conditions: ■ SB=1 (master mode); ■ ADDR=1 (master/slave mode) ■ ADD10=1 (master mode); ■ STOPF=1 (slave mode) ■ BTF=1 but no TxE or RxNE event ■ TxE event if ITBUFFEN=1 ■ RxNE event is 1 if ITBUFEN=1
8	ITERREN	RW	0	Error interrupt enable. 0: Error interrupt disabled; 1: Error interrupt is allowed; This interrupt will be generated under the following conditions: ■ BERR=1 ■ ARLO=1 ■ AF=1 ■ OVR=1 ■ PECERR=1 ■ TIMEOUT=1 ; ■ SMBAlert=1.
7:6	Reserved	RES	-	Reserved
5:0	FREQ	RW	0	I2C module clock frequency.

Bit	Name	R/W	Reset Value	Function
				This register must be configured with the value of the APB clock frequency to produce data setup and hold times compatible with the I2C protocol. The running clock is an even clock. The minimum allowable settable frequencies are 4MHz and above in 100k mode and greater than 8MHz in 400k mode. 000000: Prohibit 000001: Prohibit ..... 000011: Prohibit 000100: 4MHz ..... 110010: 50MHz Greater than 110010: Prohibit.

26.3.3. I2C Own address register1 (I2C_OAR1)

Address offset:0x08

Reset value:0x4000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDMODE	Res.	Res.	Res.	Res.	Res.	ADD[9:8]		ADD[7:1]							ADD0
						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	ADDMODE	RW	0	Addressing mode (Slave mode). 0: 7-bit slave address (Does not respond to 10-bit address); 1: 10-bit slave address (Does not respond to 7-bit address);
14:10	Reserved	RES	0	Reserved
9:8	ADD[9:8]	RW	0	Interface Address. 7-bit address mode This register is irrelevant. 10-bit address mode Bits 9 to 8 of the bit address.
7:1	ADD[7:1]	RW	0	Bits 7~1 of the interface address.
0	ADD0	RW	0	Interface Address. 7-bit address mode This register is invalid. 10-bit address mode Bit 0 of the bit address.

26.3.4. I2C own address register2 (I2C_OAR2)

Address offset:0x0C

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD2[7:1]							ENDUAL
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	Reserved	RES	-	Reserved
7:1	ADD2[7:1]	RW	0	Bits 7~1 of the interface address. Bits 7~1 of the address in dual address mode.
0	ENDUAL	RW	0	Dual address mode enable bit. 0: In 7-bit address mode, only OAR1 is recognized; 1: In 7-bit address mode, both OAR1 and OAR2 are recognized.

26.3.5. I2C Data register(I2C_DR)

Address offset:0x10

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[7:0]							
								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	Reserved	RES	-	Reserved
7:0	DR[7:0]	RW	0	8-bit data register, inside the chip is actually two independent buffer share a common address, respectively used to store the received data (RX_DR), placed to send to the bus data (TX_DR). **Transmitter mode:** When a byte is written to the DR register (actually written to TX_DR), the data transfer is automatically initiated. Once the transfer has started (TxE=1), the I2C module will maintain a continuous flow of data if the next data to be transferred is written to the DR register in time. **Receiver mode:** The received byte is copied to the DR register (actually RX_DR) (RxNE=1). Continuous data reception is achieved by reading the data register before the next byte (RxNE=1) is received. Notes:

Bit	Name	R/W	Reset Value	Function
				1) In slave mode, the address is not copied into the data register DR 2) Hardware does not handle write conflicts (if TxE=0, it can still write to the data register) 3) If an ARLO event occurs while the ACK pulse is being processed, the received byte will not be copied into the data register and therefore cannot be read

26.3.6. I2C Staus register (I2C_SR1)

Address offset:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMBALERT	TIMEOUT	Res.	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB
RC_W0	RC_W0		RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	R	R		R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
15	SMBALERT	RC_W0	0	SMBus alert status. SMBus host mode: 0: no SMBus alert; 1: SMBAlert reminder event generated at pin; SMBus slave mode: 0: no SMBAlert response address header sequence; 1: SMBAlert response address header sequence received until SMBAlert goes low. This bit is cleared by software writing 0 or by hardware when PE=0.
14	TIMEOUT	RC_W0	0	Timeout or Tlow error. 0: No timeout error; 1: The duration of SCL being low has reached 25ms (timeout); or the cumulative clock extension time of the host being low has exceeded 10ms (Tlow:next); or the cumulative clock extension time of the slave device being low has exceeded 25ms (Tlow:next). When this bit is set in slave mode: the slave device resets the communication and the hardware releases the bus;

Bit	Name	R/W	Reset Value	Function
				When this bit is set in master mode: the hardware issues a stop condition; This bit is cleared by software writing 0, or by hardware when PE=0. Note: This function is only available in SMBUS mode.
13	Reserved	RES	-	Reserved
12	PECERR	RC_W0	0	A PEC error occurs on reception. 0: No PEC error, return ACK after receiving PEC (if ACK=1) 1: There is a PEC error, and NACK is returned after receiving the PEC (regardless of what value ACK has) This bit is cleared by software writing 0, or by hardware when PE=0.
11	OVR	RC_W0	0	Overload/Underload flag. 0: No overload/underload; 1: overload/underload present. When NOSTRETCH=1, this bit is set by hardware in slave mode; In receive mode when a new byte is received (including the ACK answer pulse) and the contents of the data register have not been read out, the newly received byte will be lost. In transmit mode when a new byte is to be sent without new data being written to the data register, the same byte will be sent twice. This bit is cleared by software writing 0 or by hardware when PE=0. Note: If a write operation to the data register occurs very close to the rising edge of SCL, the data sent is indeterminate and a hold time error occurs.
10	AF	RC_W0	0	Answer failure flag. 0: No answer failure; 1: answer failure. Hardware sets this register when no answer is returned. This bit is cleared by software writing 0, or by hardware when PE=0.
9	ARLO	RC_W0	0	Arbitration loss (master mode). 0: No arbitration loss detected;

Bit	Name	R/W	Reset Value	Function
				1: Arbitration loss detected. This register is set by hardware when the interface loses control of the bus to another host. This bit is cleared by software writing 0, or by hardware when PE=0. After an ARLO event, the I2C interface automatically switches back to slave mode (M/SL=0). Note: In SMBUS mode, arbitration of data in slave mode occurs only during data decode, or answer transmission intervals (excluding answers to addresses).
8	BERR	RC_W0	0	Bus error flag. 0: No start or stop condition error; 1: start or stop condition error. When the interface detects an erroneous start or stop condition, hardware sets this bit to 1. This bit is cleared by software writing 0, or by hardware when PE=0.
7	TxE	R	0	Data register is empty (on send) flag. 0: Data register is not empty; 1: The data register is empty. This bit is set to 1 when the data register is empty when sending data, and is not set during the send address phase. Software writing data to the DR register clears this bit, or it is automatically cleared by hardware after a start or stop condition occurs, or when PE=0. This bit is not set if a NACK is received, or if the next byte to be sent is PEC (PEC=1). Note: The TxE bit cannot be cleared after writing the 1st data to be sent, or writing data when BTF is set, because the data register is empty at that time.
6	RxNE	R	0	Data register non-empty (on receive) flag. 0: Data register is empty; 1: The data register is not empty. At reception, when the data register is not empty, set this register. This register is not set during the receive address phase.

Bit	Name	R/W	Reset Value	Function
				Software read and write operations to the data register clear this register, or it is cleared by hardware when PE=0. Note: When BTF is set, reading data cannot clear the RxNE bit because the data register is still full at that time.
5	Reserved	RES	-	Reserved
4	STOPF	R	0	Stop condition detection bit (slave mode). 0: No stop bit detected; 1: Stop condition detected. After an answer (if ACK=1), hardware sets this bit 1 when the slave device detects a stop condition on the bus. A write operation to the I2C_CR1 register after a software read of the I2C_SR1 register will clear this bit, or hardware will clear this bit when PE=0. Note: The STOPF bit will not be set after a NACK is received.
3	ADD10	R	0	The 10-bit address header sequence has been sent (master mode). 0: No ADD10 event has occurred. 1: The master device has sent the first address byte out. In 10-bit address mode, when the master device will have sent the first byte out, the hardware sets this position 1. A write operation to the I2C_DR register after software reads the I2C_SR1 register will clear this bit, or hardware clears this bit when PE=0. Main: This register is not set after a NACK is received.
2	BTF	R	0	Byte transmission send end flag bit. 0: Byte transmission not completed 1: Byte transmission send successfully finished When NOSTRETCH=0, the hardware will set this register in the following cases (when slave mode, when NOSTRETCH=0; master mode, independent of NOSTRETCH): ■ When receiving, when a new byte is received (including the ACK pulse) and the data register has not yet been read (RxNE=1).

Bit	Name	R/W	Reset Value	Function
				■ When transmitting, when a new data should be sent and the data register has not yet been written to with the new data (TxE=1). This bit is cleared by a read or write operation to the data register after the I2C_SR1 register has been read by software, or by hardware after a start or stop condition has been sent, or when PE=0. Notes: The BTF bit is not set after a NACK is received. If the next byte to be transmitted is a PEC (I2C_SR2.TRA=1,I2C_CR1.PEC=1), the BTF bit will not be set.
1	ADDR	R	0	Address has been sent (master mode)/address match (slave mode). A software read of the I2C_SR2 register followed by a read or write operation to the data register from the I2C_SR1 register will clear this bit; or it will be cleared by hardware after a start or stop condition is sent in a transmission, or when PE=0. **Address Match (Slave):** 0: Address mismatch or no address received; 1: Received address match. This bit is set by hardware when the received slave address matches the OAR register or general call address, or the default address of the SMBus device, or when the SMBus host recognizes an ex-SMBus alert. **Note:** In slave mode, a full clear sequece is recommended, i.e., after ADDR is set, the SR1 register is read first, then the SR2 register. **Address sent (Master):** 0: Address sending has not ended; 1: address sending is finished. ■ For 10-bit addresses, the bit is set when an ACK is received for the second byte of the address; ■ For 7-bit addresses, set when the ACK byte of the address is received. **Note:** This register will not be set after a NACK is received.
0	SB	R	0	Start bit flag (master mode).

Bit	Name	R/W	Reset Value	Function
				0: start condition not sent; 1: the start condition has been sent; ■ Sets this register when the start condition is sent. ■ After software reads the I2C_SR1 register, a read or write operation to the data register will clear this bit; or when PE=0, it will be cleared by hardware.

26.3.7. I2C Status register2 (I2C_SR2)

Address offset:0x18

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Res.	TRA	BUSY	MSL
R	R	R	R	R	R	R	R	R	R	R	R		R	R	R

Bit	Name	R/W	Reset Value	Function
15:8	PEC	R	0	Packet Error Detection Register. When ENPEC=1, this register holds the value of the internal PEC.
7	DUALF	R	0	Dual address flag (slave mode) . 0: Received address matches OAR1; 1: the received address matches OAR2. Hardware clears this register when a stop condition or a repeated start condition is generated, or when PE=0.
6	SMBHOST	R	0	Received SMBus host header sequence (slave mode) flag. 0: No SMBus host address received; 1: SMBus host address received when SMBTYPE=1 and ENARP=1. Hardware clears this register when a stop condition or a repeated start condition is generated, or when PE=0.
5	SMBDEFAULT	R	0	Default address of the SMBus slave device (slave mode). 0: The default address of the SMBus device was not received; 1: Default address of SMBus device received when ENARP=1.

Bit	Name	R/W	Reset Value	Function
				Hardware clears this register when a stop condition or a repeated start condition is generated, or when PE=0.
4	GENCALL	R	0	Broadcast call address (slave mode). 0: Broadcast call address not received; 1: Broadcast call address received when ENGC=1. Hardware clears this register when a stop condition or a repeated start condition is generated, or when PE=0.
3	Reserved	RES	-	Reserved
2	TRA	R	0	Send/receive flag. 0: Data received 1: Data sent At the end of the entire address transfer phase, this register is set according to the R/W bit of the address byte. Hardware clears this register when a stop condition is detected (STOPF=1), or a repeated start condition, or a bus arbitration loss (ARLO=1), or when PE=0.
1	BUSY	R	0	Bus busy flag. 0: No data communication on the bus 1: Polarized data communication is in progress on the bus Hardware set when SDA or SCL is detected as low. Hardware cleared when a stop condition is detected. This register indicates the current bus communication in progress, and this information is still updated when the interface is disabled (PE=0).
0	MSL	R	0	Master-slave mode. 0: slave 1: master -Hardware set when the interface is in master mode (SB=1); -Hardware cleared when a stop condition is detected on the bus (STOPF=1), when arbitration is lost (ARLO=1), or when PE=0.

26.3.8. I2C Clock control register(I2C_CCR)

Address offset:0x1c

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Res.	Res.	CCR[11:0]											
RW	RW			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15	F/S	RW	0	I2C master mode selection. 0: Standard mode 1: Fast mode
14	DUTY	RW	0	Duty cycle in fast mode. 0: In fast mode: $T_{low}/T_{high}=2$ 1: In fast mode: $T_{low}/T_{high}=16/9$
13:12	Reserved	RES	-	Reserved
11:0	CCR[11:0]	RW	0	Clock control crossover factor in fast/standard mode (master mode). This division factor is used to set the SCL clock in master mode. ■ Standard mode or SMBus mode: - T_{high}=CCR x Tpclk - T_{low} =CCR x Tpclk ■ Fast mode: - DUTY=0: 1. T_{high}=CCR x Tpclk 2. T_{low} =2 x CCR x Tpclk - DUTY=1(Reach 400KHz): 3. T_{high}=9 x CCR x Tpclk 4. T_{low} =16 x CCR x Tpclk Note: 1. The minimum value allowed to be set is 0x04, and the minimum value allowed in fast DUTY mode is 0x01 2. $T_{high}=t_{r(SCL)}+t_{w(SCLH)}$ 3. $T_{low}=t_{r(SCL)}+t_{w(SCLL)}$ 4. There is no filter for these delays 5. This register can only be configured when PE=0;

26.3.9. I2C TRISE register (I2C_TRISE)

Address offset:0x20

Reset value:0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRISE[5:0]					

										RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:6	Reserved	RES	-	Reserved
5:0	TRISE	RW	0x0002	Maximum rise time in fast/standard mode (master mode). These bits should provide the maximum duration of the SCL feedback loop in master mode. The purpose of this is to keep the SCL at a stable frequency regardless of the duration of the SCL rising edge. These bits must be set to the maximum SCL rise time given in the I2C bus specification in steps of one. For example, the maximum allowable SCL rise time in standard mode is 1000ns. if the value in FREQ[5:0] in the I2C_CR2 register is equal to 0x08 and Tpclk = 125ns, the configuration in TRISE is 0x09 (1000ns/125ns = 8 + 1 = 9). The value of the filter can also be added inside TRISE. If the result is not an integer, the integer portion is written to TRISE to ensure the tHIGH parameter. Note: This register can only be set when PE = 0.

26.3.10. I2C register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Res.	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Res.	Res.	FREQ[5:0]					
	Reset value																				0	0	0	0	0			0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDMODE	Res.	Res.	Res.	Res.	Res.	ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	I2C_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	I2C_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMBALERT	TIMEOUT	Res.	PECERR	OVR	AF	ARLO	BERR	TXE	RXNE	Res.	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	I2C_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Res.	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0
0x1C	I2C_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F/S	DUTY	Res.	Res.	CCR[11:0]											
	Reset value																	0	0			0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	TIM1_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRISE[5:0]					
	Reset value																											0	0	0	0	1	0

27. Universal synchronous asynchronous receiver transmitter (USART)

27.1. Introduction

The Universal Synchronous Asynchronous Transceiver (USART) provides a flexible method of exchanging full-duplex data with external devices using the industry-standard NRZ asynchronous serial data format.The USART utilizes a fractional baud rate generator to provide a wide range of baud rate options.

It supports synchronous unidirectional and half-duplex single-wire communications, as well as LIN (Local Internet), Smart Card Protocol and IrDA (Infrared Data Organization) ENDEC specifications, and modem (CTS/RTS) operation. It also allows multiprocessor communication.

High-speed data communication is possible using the DMA method with multi-buffer configuration.

27.1.1. Main features

27.1.1.1. USART Features

- Full duplex asynchronous communication
- Programmable baud rate up to 4.5 Mbit/s common to transmit and receive
- Configurable data word length (8 or 9 bits)
- Configurable stop bits - 0.5, 1, 1.5 or 2 stop bits are supported.
- Transmitter provides clock for synchronous transmission
- Single-wire half-duplex communication
- Separate transmitter and receiver enable bits
- Parity control
- Transmit parity bit
- Verification of received data
- Detection Flag
- Receive buffer full
- Transmit buffer empty
- End of transmission flag
- Ability for Lin masters to send synchronized disconnect symbols and Lin slaves to detect disconnect symbols
- When the USART hardware is configured for LIN, a 13-bit disconnect is generated and a 10/11 disconnect is detected
- IRDA SIR Encoder Decoder
- Supports 3/16-bit duration in normal mode
- Smart Card Analog Function
- The smart card interface supports the asynchronous smart card protocol defined in the ISO7816-3 standard.
- 0.5 and 1.5 stop bits used by smart cards
- Four False Detection Signs
- Overflow Error
- Noise Error

- Framing errors
- Checksum Error
- 10 interrupt sources with flags
- CTS change
- LIN break character detection
- Transmit data register is empty
- Transmit complete
- Receive data register full
- Bus idle detected
- Overflow error
- Frame error
- Noise error
- Checksum error
- Multiprocessor communication. If the addresses do not match, silent mode is entered
- Wake-up from silent mode (by idle bus detection or address flag detection)
- Two ways to wake up the receiver: address bit (MSB, bit 9), bus free
- Support 8-bit and 16-bit oversampling modes
- Full-duplex asynchronous, half-duplex asynchronous, multiprocessor communication (belongs to full-duplex asynchronous): supports configuration of 8-bit and 16-bit sampling;
- Full-duplex synchronous, LIN, smart card, infrared: 16-bit sampling by default, no need to configure OVER8 bits of USART_CR3. Synchronization does not take into account how many bits are sampled.

27.1.1.2. USART Feature Description

The interface is connected to other devices via three pins (see Figure 248). At least two pins are required for any USART bidirectional communication: receive data input (RX) and transmit data output (TX).

RX: Receive data serial input. Data is recovered by an oversampling technique to distinguish data from noise.

TX: Transmit data output. When the transmitter is disabled, the output pin reverts to its I/O port configuration. When the transmitter is activated and no data is sent, the TX pin is high. In single-wire and smart card modes, this I/O port is used for both sending and receiving data.

- The bus should be idle before transmitting or receiving
- One start bit
- A data word (8 or 9 bits) with the least significant bit first.
- 0.5, 1.5, and 2 stop bits, which indicate the end of the data frame.
- Using a fractional baud rate generator - 12-bit integer and 4-digit decimal representation.
- A status register (USART_SR)
- A data register (USART_DR)
- A baud rate register (USART_BRR), 12-bit integer and 4 decimal bits
- A smart card mode guard time register (USART_GTPR)

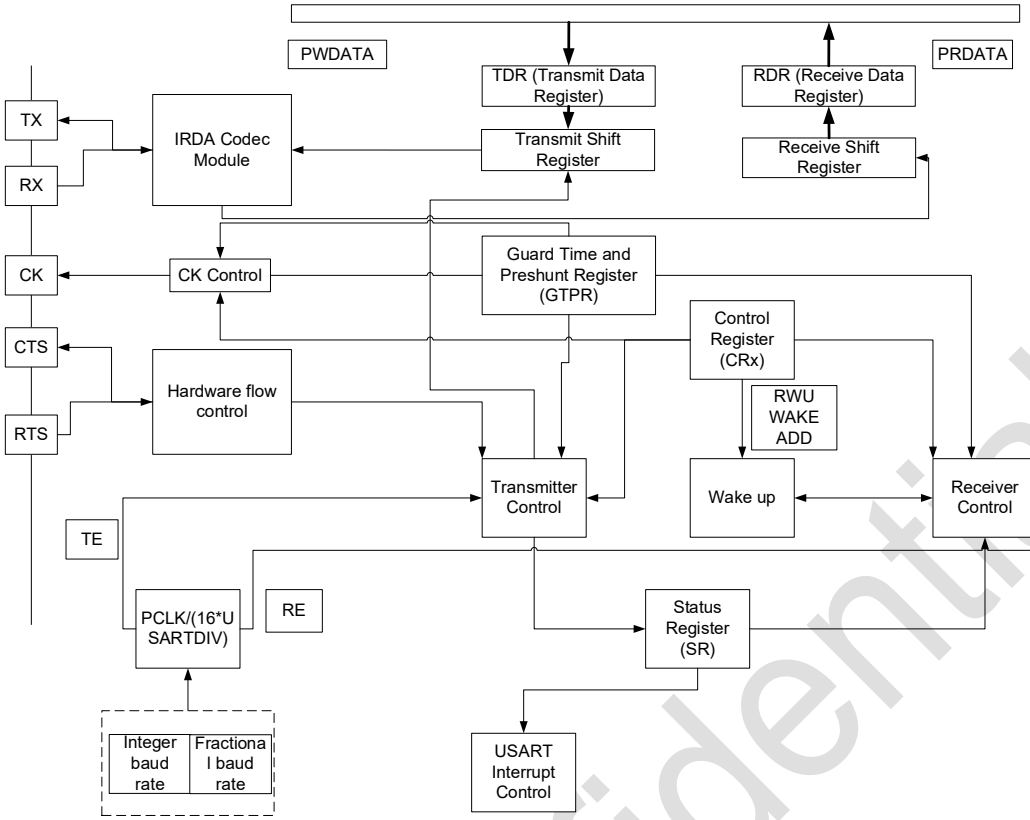


Figure 27-1 USART Block Diagram

The word length can be selected as 8 or 9 bits by programming the M bit in the USART_CR1 register (see Figure 1.1). The TX pin is low during the start bit and high during the stop bit.

The idle symbol is considered to be a complete data frame consisting entirely of '1's followed by the start bit of the next frame containing the data (the number of '1' bits also includes the number of stop bits).

The break symbol is considered to have received all '0's (including the stop bit period, which is also a '0') in one frame cycle. At the end of the break frame, the transmitter inserts 1 or 2 more stop bits ('1') to answer the start bit.

Transmission and reception are driven by a common baud rate generator, which generates separate clocks for the transmitter and receiver when their respective enable bits are set.

27.2. USART Functional Description

27.2.1. Transmission Pin

27.2.1.1. Basic transmission pin

Any USART bidirectional communication requires at least two pins: receive data input (RX) and transmit data output (TX).

RX: Receive data serial input. Data is recovered by an oversampling technique to distinguish between data and noise.

TX: Transmit data output. When the transmitter is disabled, the output pin reverts to its I/O port configuration. When the transmitter is enabled, but not sending data, the TX pin is high. In single-wire mode, this I/O port is used for both sending and receiving data.

27.2.1.2. Mode pin

Add CK pin in synchronized mode.

CK: Transmitter clock output.

This pin outputs the clock used to synchronize the transmission, (there is no clock pulse on the Start and Stop bits, software optionally, a clock pulse can be sent on the last data bit). Data can be received synchronously on RX. This can be used to control external devices with shift registers. The clock phase and polarity are software programmable.

27.2.1.3. Hardware flow control mode pin

CTS: low to start transmitting, end of transmitting becomes high.

RTS: a low level indicates that the USART is ready to receive data.

27.2.2. USART Transmit

The transmitter sends an 8-bit or 9-bit data word depending on the state of the M bit. When the transmit enable bit (TE) is set, the data in the transmit shift register is output on the TX pin and the corresponding clock pulse is output on the CK pin.

Character Transmission

During USART transmit, the least significant bit of data is shifted first on the TX pin. In this mode, the USART_DR register contains a buffer between the internal bus and the transmit shift register (see Figure 1-1).

Each character is preceded by a low start bit; this is followed by a configurable number of stop bits. The USART supports multiple stop bit configurations: 0.5, 1, 1.5, and 2 stop bits.

Notes:

1) The TE bit cannot be reset during a data transfer, otherwise the data on the TX pin will be corrupted because the baud rate counter stops counting. The current data being transmitted will be lost.
2) An idle frame will be sent when the TE bit is activated.

27.2.2.1. Transmission step

- Configure USART_CR1.M to determine whether to transmit 8 bits or 9bits;
- Configure the USART_BRR register to select the baud rate;
- Configure the USART_CR2.STOP register to define the number of stop bits.
- Configure USART_CR1.UE=1 to enable USART.
- Configure USART_CR3.DMAT=1 to enable DMA when multiprocessor transmission is in progress.
- Configure TE=1 (TE=1 is required before writing data to the USART_TDR register and TE is maintained high throughout the transfer phase).
- Hardware insertion Idle timing
- Write transfer data to the USART_TDR register (this operation clears the TXE bit). This operation may be repeated several times.
- Hardware sets TXE to indicate that the USART_TDR register data has been moved to the shift register and that the USART_TDR register is empty and new data can be written.
- Hardware generates the transmit timing:
    - Start bit, start transmission

- Transmit shift register output data (default LSB) to TX pin, CK pin of clock signal output (synchronization mode).

Stop bit (the number of transmissions is configured).

- TC=1 is generated after the last frame is transmitted (when TXE=1). TC interrupt is generated when TCIE=1.

27.2.2.2. TC/TXE Timing

Clearing the TXE bit is always accomplished by a write operation to the data register. The TXE bit is set by hardware, which indicates:

- Data has been shifted from the TDR to the shift register, data transmission has started
- TDR register is cleared

The next data can be written to the USART_DR register without overwriting the previous data

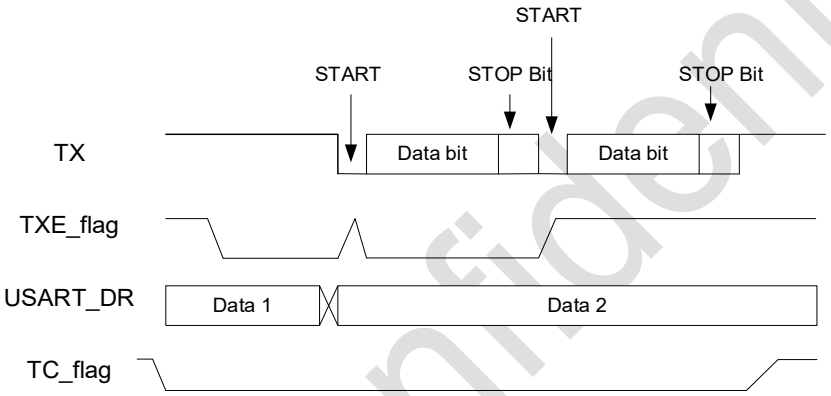


Figure 27-2 Changes in TC/TXE during USART transmission

Disconnect Symbol

Setting SBK sends a break symbol. The disconnect frame length depends on the M bit. If SBK=1 is set, a break symbol will be sent on the TX line after completion of the current data transmission. SBK is hardware reset when the disconnect character is sent (at the stop bit of the disconnect symbol). The USART inserts a logical '1' at the end of the last break frame to ensure that the start bit of the next frame is recognized.

Note: If the software resets the SBK bit again before starting to send a break frame, the break symbol will not be sent. If two consecutive break frames are to be sent, the SBK bit should be set after the stop bit of the previous break symbol.

27.2.3. Data reception

27.2.3.1. Start Bit Detection

In USART, a start bit is considered to be detected if a special sampling sequence is recognized. The sequence is: 1110X0X0X0000. note: If the sequence is incomplete, then the receiver will exit start bit detection and return to the idle state (no flag bits set) waiting for a falling edge. If all 3 samples are 0 (the first sample at bits 3, 5, and 7, and the second sample at bits 8, 9, and 10), then receipt of the start bit is acknowledged, at which point the RXNE flag bit is set and an interrupt is generated if RXNEIE=1. If only 2 of the 3 samples are 0 on both occasions (the samples of bits 3, 5, and 7 and the samples of bits 8, 9, and 10), then the start bit is still valid, but the NE noise flag bit is set. If this condition cannot be met, the start bit detection process is aborted and the receiver returns to the idle state (no flag bit is set).

If there is a time when only 2 of the 3 sample points are 0 (sample points of bits 3, 5, and 7 or sample points of bits 8, 9, and 10), then the start bit is still valid, but the NE noise flag bit is set.

27.2.3.2. Character reception

Character Receive Configuration Steps:

- Configure USART_CR1.M to select the receive length;
- Configure the USART_BRR register to select the baud rate;
- Configure USAR_CR2.STOP, 1bit or 2bit stop bit;
- Configure USART_CR1.UE=1 to enable USART;
- Configure USART_CR3.DMAR=1 to enable DMA for multiprocessor transfers;
- Configure USART_CR1.RE to enable reception and start detecting the start bit;

When 1 character is received:

- Setting RXNE. indicates that the contents of the shift register have been transferred to RDR, i.e. the data can be read;
- When RXNEIE=1, an interrupt is generated;
- On reception, if a framing error, noise or overflow error is detected, the corresponding error flag bit is set;
- For multiprocessor communication, RXNE is set after each byte is received and cleared by a DMA read operation to the receive data register;
- In single-buffer mode, software reads the USART_RDR register to complete clearing of the RXNE bit.The RXNE flag can also be cleared by writing USART_RQR.RXFRQ=1. The RXNE bit must be cleared before the end of the next character reception to avoid overflow errors.

Note: The RE bit should not be reset while receiving data. If the RE bit is cleared on reception, reception of the byte is lost at that time.

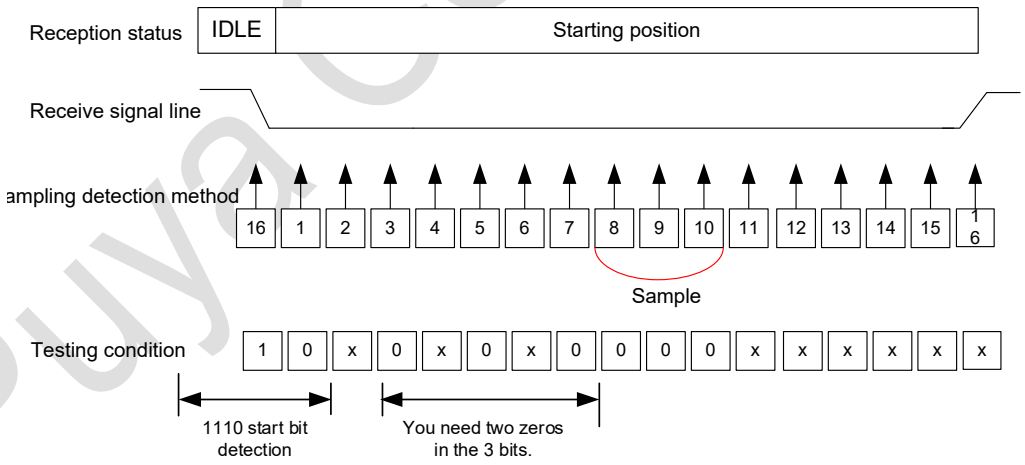


Figure 27-3 USART start bit detection

Special frames and error handling:

- Break frame: when a break frame is received, the USART processes it as an error frame.
- Idle frame: when an Idle frame is detected, it is handled as a normal frame and an interrupt is generated if IDLEIE=1.
- Overrun error: If RXNE is not reset and another character is received, an overrun error occurs. Data can be transferred from the shift register to the RDR register only when the RXNE bit is cleared. the RXNE flag is set after each byte is received. If the RXNE flag remains set if the next

data has been received or if a previous DMA request has not been executed, an overflow error is generated.

The following error handling is performed when an overflow error is generated:

- Setting the ORE;
- RDR contents are not lost. Reading the USART_DR register still reads the previous data normally;
- Shift register contents are overwritten and subsequently received data is lost;
- If RXNEIE=1, or EIE=1, an interrupt is generated;
- Write ORECF=1 clears the ORE bit;

ORE is set, indicating that at least 1 piece of data has been lost. There are two scenarios as follows:

- If RXNE=1, the last valid data is still in the receive register RDR and can be read;
- If RXNE=0, the last valid data has been read away and there is no valid data to read in RDR. This situation may occur when the last valid data is read in RDR while new (lost) data is received.

Noise Error:

Data recovery using oversampling techniques (except in synchronized mode) by distinguishing between valid in put data and noise.

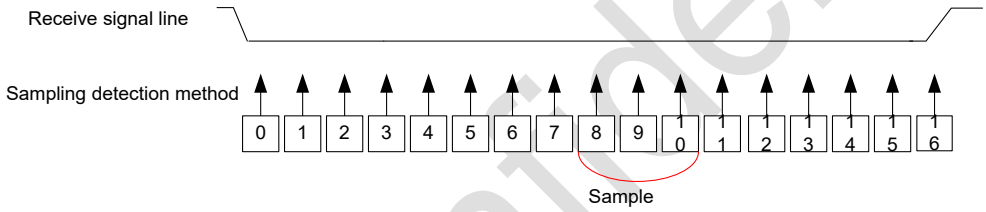


Figure 27-4 Data sampling for noise detection

Table 27-1 Data sampling for noise detection

Sampling value	NE status	Receive data	Data validity
000	0	0	Validity
001	1	0	Null
010	1	0	Null
011	1	1	Null
100	1	0	Null
101	1	1	Null
110	1	1	Null
111	0	1	Validity

When noise is detected in a received frame:

- The NE flag is set on the rising edge of the RXNE bit.
- Invalid data is transferred from the shift register to the USART_DR register.
- In the case of single byte communication, no interrupt is generated. However, since the NE flag bit and the RXNE flag bit are set at the same time, RXNE will generate an interrupt. In the case of multi-buffer communication, if the EIE bit in the USART_CR3 register has been set, an interrupt will be generated.

Reading the USART_SR and then the USART_DR registers will clear the NE flag bit.

Frame Error:

A frame error is detected when. the stop bit is not picked up and received recognized at the expected time due to not being synchronized on or a large amount of noise.

When a frame error is detected:

- FE bit is set by hardware
- Invalid data is transferred from the shift register to the USART_DR register.
- During single-byte communication, no interrupt is generated. However, this bit is set at the same time as the RXNE bit, which generates an interrupt. In the case of multi-buffer communication, an interrupt is generated if the EIE bit in the USART_CR3 register is set.

Sequential execution of read operations to the USART_SR and USART_DR registers can reset the FE bit.

Configurable stop bits during reception.

The number of stop bits to be received can be configured by the control bits in control register 2, which can be 1 or 2 in normal mode, and may be 0.5 or 1.5 in smart card mode.

- 0.5 stop bits (reception in smart card mode): 0.5 stop bits are not sampled. Therefore, frame errors and break frames cannot be detected if 0.5 stop bits are selected.
- 1 stop bit: Sampling of 1 stop bit is performed on the 8th, 9th and 10th sampling points.
- 1.5 stop bits (smart card mode): When transmitting in smart card mode, the device must check that the data has been sent correctly. So the receiver function block must be activated (RE = 1 in the USART_CR1 register) and the signal on the data line is sampled during the sending of the stop bit. If a checksum error occurs, the smart card indicates a framing error by pulling down the data line when the sender samples the NACK signal, i.e., during the time corresponding to the stop bit on the bus. FE is set at the end of the 1.5 stop bits along with RXNE. Sampling of the 1.5 stop bits is performed at sample points 16, 17 and 18. The 1.5 stop bits can be divided into 2 parts: one is 0.5 clock cycles during which nothing is done. thing . This is followed by 1 clock cycle of stop bits, sampled at the midpoint of this period.
- 2 stop bits: Sampling of the 2 stop bits is done at the 8th, 9th and 10th sample points of the first stop bit. If a frame error is detected during the first stop bit, the frame error flag is set. The second stop bit is no longer checked for frame errors. The RXNE flag will be set at the end of the first stop bit.

27.2.4. Fractional baud rate generation

The baud rates of the receiver and transmitter should be set to the same values in the integer and decimal registers of the USARTDIV.

$$\text{Tx/Rx baud} = \frac{fck}{(16 * USARTDIV)}$$

Here f_{CK} is the clock given to the peripheral (PCLK1 is used for USART2, 3, 4, 5 and PCLK2 is used for USART1) USARTDIV is an unsigned fixed point number. The value of these 12 bits is set in the USART_BRR register.

Note: After writing USART_BRR, the baud rate counter is replaced by the new value of the baud rate register. Therefore, do not change the value of the baud rate register while communication is in progress.

Table 27-2 Error calculation when setting baud rate

Baud rate		Fpclk = 8MHZ			Fpclk = 36MHZ			Fpclk = 100MHZ		
No.	kbps	Reality	Value placed in baud rate register	Inaccuracy %	Reality	Value placed in baud rate register	Inaccuracy %	Reality	Value placed in baud rate register	Inaccuracy %
1	2.4	2.400	208.375	0.02%	2.400	937.5	0.00%	2.399998	2604.1875	0.00%
2	9.6	9.604	52.0625	0.04%	9.600	234.375	0.00%	9.59417	651.4375	0.06%
3	19.2	19.198	26.04375	0.01%	19.200	117.1875	0.00%	19.2012	325.5000	0.01%
4	57.6	57.554	8.6875	0.08%	57.600	39.0625	0.00%	57.6037	108.5000	0.01%
5	115.2	115.942	4.3125	0.64%	115.385	19.5	0.16%	115.207	54.2500	0.01%
6	230.4	228.571	2.1875	0.80%	230.769	9.75	0.16%	230.415	27.1250	0.01%
7	460.8	470.588	1.0625	2.08%	461.538	4.875	0.16%	460.829	13.5625	0.01%
8	921.6		Impossible		923.077	2.4375	0.16%	925.926	6.7500	0.47%
9	2250		Impossible		2250.000	1	0.00%	2272.73	2.7500	1.00%
10	4500		Impossible	Impossible	Impossible	Impossible	Impossible	4545.45	1.3750	1.00%

Notes:

1) The lower the clock frequency of the CPU, the lower the error for a particular baud rate. The upper limit of the achievable baud rate can be obtained from this set of data.
2) Only USART1 uses PCLK2. other USARTs use PCLK1.
3) When configuring the clock, the baud rate clock is divided by the system clock, but the clock frequency obtained will be in error from the actual clock frequency. The configured clock error needs to be within 2% to work properly. When the fpclk is at 100MHZ, the baud clock cannot be set greater than 4.5MHZ.

27.2.5. USART receiver tolerates clock changes

The USART Asynchronous Receiver will operate properly only if the overall clock system varies less than the USART Asynchronous Receiver can tolerate. Factors that affect these variations are:

- DTRA: variations due to transmitter errors (including variations in the oscillator at the transmitter side)
- DQUANT: error due to baud rate rounding at the receiver side
- DREC: variation of the oscillator at the receiver side
- DTCL: variations due to the transmission line (usually caused by the inconsistency between the transceiver's low-to-high conversion timing and the high-to-low conversion timing).

Needs to be satisfied: DTRA + DQUANT + DREC + DTCL < USART Receiver Tolerance

27.2.6. USART automatic baud rate detection

- **Automatic baud rate detection**

The USART can automatically detect the baud rate based on the received characters and configure the baud rate register USART_BRR.

Automatic baud rate detection applies in the following situations:

The system communication speed is unknown in advance;

The system uses a less accurate clock source, a mechanism that allows the correct baud rate to be obtained without measuring clock deviations;

The clock source frequency must be compatible with the desired communication speed (the baud rate for oversampling 16 must be between fCK/65535 and fCK/16).

Configure the automatic baud rate detection mode via USART_CR2.ABRMOD. The baud rate needs to be detected several times, each time the measured value is compared with the previous one.

- **Automatic baud rate detection mode**

Mode0: This mode is for the case where the receive byte starts with 1. In this case, the start bit to bit1 (falling edge to rising edge) is measured.

Mode1: This mode is for any character that starts with a 10xx bit pattern. In this case, the USART measures the duration of the start and the first data bit. The measurement is performed along the falling edge to the falling edge, ensuring better accuracy in the case of slow signal skew.

Parallel data mode: a check performed for each intermediate transition on the RX line. If a transition on the RX is not sufficiently synchronized with the receiver (the receiver calculates the baud rate based on 0 bits), an error is generated.

27.2.7. Multiprocessor communication

Multi-processor communication (connecting several USARTs in a network) is possible through USART. For example, a USART device can be the master, and its TX output is connected to the RX inputs of other USART slave devices; the TX outputs of the respective USART slave devices are logically linked together and connected to the RX inputs of the master device.

In a multiprocessor configuration, it is often desirable to have only the addressed receivers activated to receive subsequent data, thus reducing the excess USART service overhead introduced by the involvement of unaddressed receivers. Unaddressed devices can be enabled to be placed in silent mode with their silent function.

In silent mode:

- Any receive status bits are not set.
- All receive interrupts are disabled.
- The RWU bit in the USART_CR1 register is set to 1. The RWU can be controlled automatically by hardware or written by software under certain conditions.

Depending on the state of the WAKE bit in the USART_CR1 register, the USART can enter or exit silent mode in two ways.

- If the WAKE bit is reset: idle bus detection is performed.
- If the WAKE bit is set: an address tag detection is performed.

27.2.7.1. Idle bus detection (WAKE=0)

When the RWU bit is written 1, the USART enters silent mode. It is woken up when an idle frame is detected. The RWU is then cleared by hardware, but the IDLE bit in the USART_SR register is not set. The RWU can also be written by software to 0. The following figure gives an example of using idle bus detection to wake up and enter silent mode.

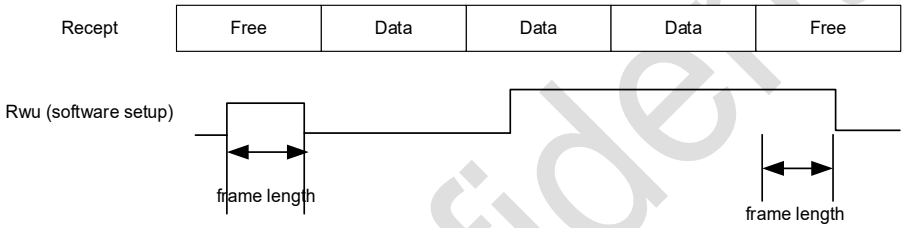


Figure 27-5 Silent mode utilizing idle bus detection

27.2.7.2. Address mark detection (WAKE=1)

In this mode, if the MSB is 1, the byte is considered as an address, otherwise it is considered as data. In an address byte, the address of the target receiver is placed in 4 LSBs. This 4-bit address is compared by the receiver to its own address, which is programmed in the ADD of the USART_CR2 register.

If the received byte does not match its programmed address, the USART enters silent mode. At this point, the hardware sets the RWU bit. Receiving this byte neither sets the RXNE flag nor generates an interrupt or issues a DMA request because the USART is already in silent mode.

When the received byte matches the address programmed in the receiver, the USART exits silent mode. The RWU bit is then cleared and the subsequent byte is received normally. Receipt of this matching address byte sets the RXNE bit because the RWU bit is cleared. When the receive buffer contains no data (RXNE=0 for USART_SR), the RWU bit can be written 0 or 1. Otherwise, this write operation is ignored. The following figure gives an example of using address mark detection to wake up and enter silent mode.

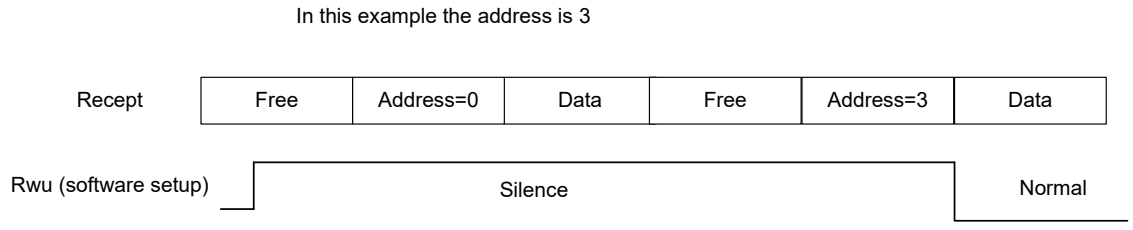


Figure 27-6 Silent mode using address tag detection

27.2.8. Calibration control

Setting the PCE bit on the USART_CR1 register enables parity control (generates a parity bit on transmit and parity check on receive). The possible USART frame formats, based on the frame length defined by the M bit, are listed in the table below.

Table 27-3 Frame format

M Bit	PCE Bit	USART Frame
0	0	|Start Bit|8-bit Data|Stop Bit|
0	1	|Start Bit|7 Bit Data|Parity Check Bit|Stop Bit|
1	0	|Start Bit|9 Bits of Data|Stop Bit|
1	1	|Start Bit|8-bit Data|Parity Check Bit|Stop Bit|

Even parity: The parity bit makes the number of 7 or 8 LSB data in a frame and the number of 1's in the parity bit even. Example: Data = 0011 0101 with 4 1's, if even parity is selected (PS = 0 in USART_CR1), the parity bit will be 0.

Odd parity: This parity bit makes the number of 7 or 8 LSB data in a frame and the number of 1's in the parity bits odd.

Example: data = 0011 0101 with 4 1's, if odd parity is selected (PS = 1 in USART_CR1), the parity bit will be 1.

Transmission mode: If the PCE bit of USART_CR1 is set, the MSB bit of the data written to the data register is replaced by the parity bit and sent out (even parity even number of 1's if even parity is selected, odd number of 1's if odd parity is selected.) If the parity check fails, the PE flag in the USART_SR register is set to 1 and an interrupt is generated if the PEIE of the USART_CR1 register is set before the parity check is set. preset, an interrupt is generated.

27.2.9. LIN (local area network) model

Configure USART_CR2.LINEN=1 to select LIN mode. In LIN mode, the lower column bit must hold bit 0:

- CLKEN of the USART_CR2 register.
- STOP/SCEN/HDSEL/IREN of the USART_CR3 register.

27.2.9.1. Transmission

Compared with normal USART transmission, LIN transmission has the following differences:

M=0. the data length is 8bit;

It is necessary to configure USART_CR2.LINEN=1. 13bit 0 will be sent as break frame after setting SBK. A bit "1" is then sent to allow detection of the next START bit.

27.2.9.2. Reception

When LIN mode is enabled, the disconnect symbol detection circuit is activated. This detection is completely independent of the USART receiver. The disconnect symbol is detected as soon as it appears, either when the bus is idle or during the sending of a data frame, which has not yet been completed, and the sending of the disconnect symbol is inserted.

When the receiver is activated (RE=1 for USART_CR1), the circuit monitors the start signal on RX. The method of monitoring the start bit is the same as detecting the break symbol or data. When the start bit is detected, the circuit samples each following bit at the 8th, 9th, and 10th oversampling

clock points of each bit. If 10 (when LBDL of USART_CR2 = 0) or 11 (when LBDL of USART_CR2 = 1) consecutive bits are 0 and followed by a delimiter, the LBD flag of USART_SR is set. If the LBDIE bit = 1, an interrupt is generated. Check the delimiter before acknowledging the disconnect symbol because it implies that the RX line has returned to high.

If a 1 is sampled before the 10th or 11th sample point, the detection circuit cancels the current detection and re-searches for the start bit. If LIN mode is disabled, the receiver continues to operate as a normal USART without regard to detecting the disconnect symbol.

If LIN mode is not activated (LINEN=0), the receiver continues to operate normally in USART mode and no disconnect detection is performed.

If LIN mode is activated (LINEN=1), as soon as a framing error occurs (i.e., a stop bit detects a '0', which occurs in a break frame), the receiver stops until the break symbol detection circuitry receives a 1 (which occurs when the break symbol has not been sent out in its entirety), or a delimiter (This occurs when a complete break symbol has been detected).

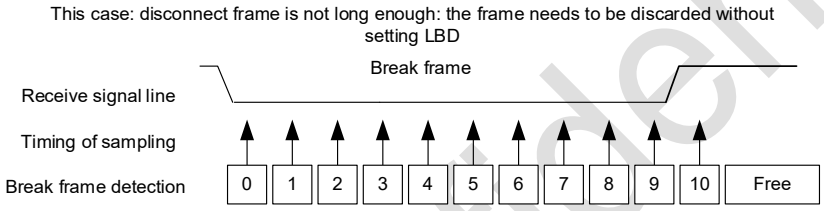


Figure 27-7 Insufficient disconnect frame length in LIN mode

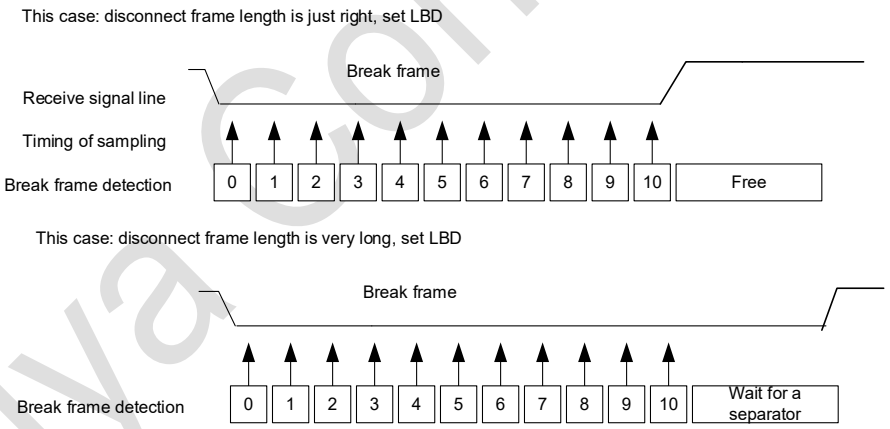


Figure 27-8 LIN mode disconnect frame length enough case

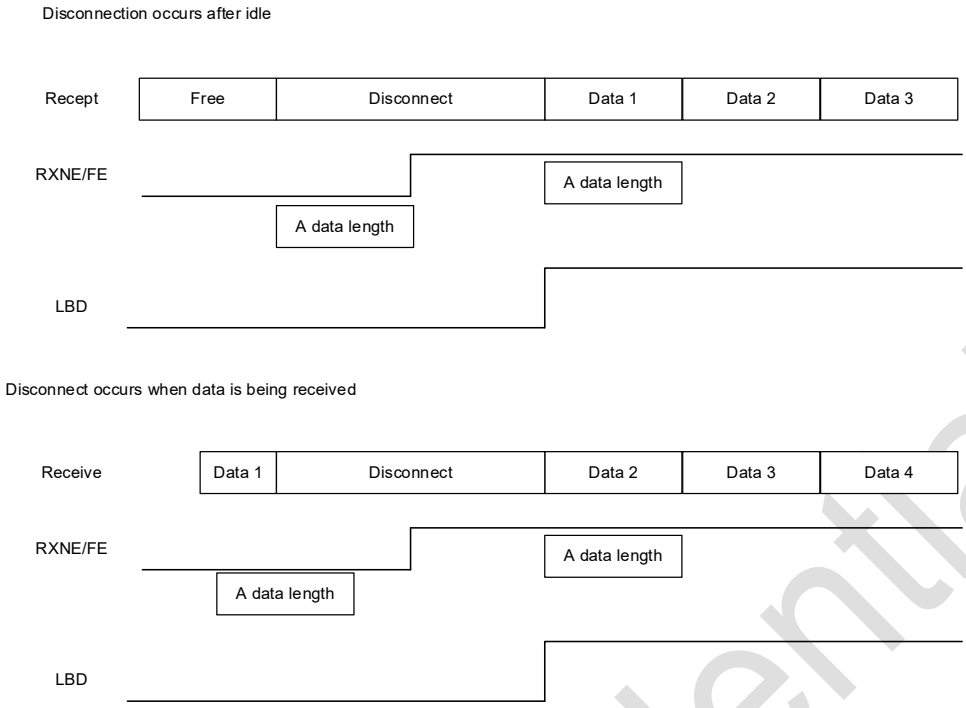


Figure 27-9 Disconnect detection and frame error detection in LIN mode

27.2.10. Synchronization Mode

Synchronization mode is selected by writing the CLKEN bit to the USART_CR2 register.

In synchronization mode, the lower column bits must be kept clear:

- LINEN bit in the USART_CR2 register
- SCEN, HDSEL, and IREN bits in the USART_CR3 register

USART allows the user to control bi-directional synchronous serial communications in a master mode. The CK pin is the output of the USART transmitter clock. There are no clock pulses on the CK pin during the start and stop bits. Depending on the state of the LBCL bit in the USART_CR2 register, it is decided to generate or not generate a clock pulse during the last valid data bit. The CPOL bit in the USART_CR2 register allows the user to select the clock polarity and the CPHA bit on the USART_CR2 register allows the user to select the phase of the external clock.

The external CK clock is not activated during bus idle, before the actual data arrives, and when the break symbol is sent. In synchronous mode, the USART transmitter works exactly as in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is sent synchronously with CK.

The USART receiver in synchronous mode works differently than in asynchronous mode. If RE=1, data is sampled on CK (on the rising or falling edge according to CPOL and CPHA) without any oversampling. However, the build-up time and duration (depending on the baud rate, 1/16 bit time) must be taken into account.

Notes:

1) The CK pin works jointly with the TX pin. Consequently, the clock is only provided when the transmitter is enabled (TE = 1) and data is sent (written to the USART_DR register). This means that it is not possible to receive a synchronized data when no data is sent.

2) The correct configuration of the LBCL, CPOL and CPHA bits should be when both the transmitter and receiver are disabled; these bits cannot be changed when either the transmitter or receiver is enabled.

3) It is recommended that TE and RE be set in the same instruction to minimize receiver setup time and hold time.

4) USART only supports master mode: it cannot receive or send data with an input clock from another device (CK is always an output).

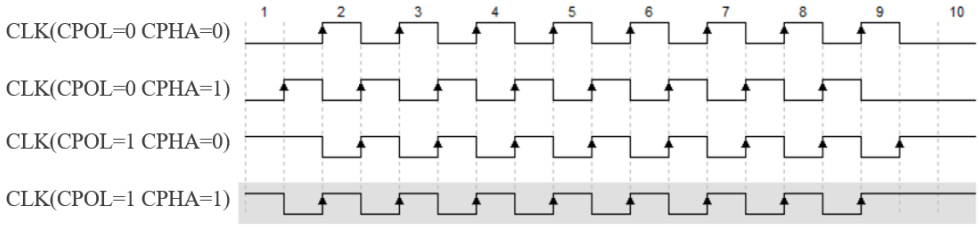


Figure 27-10 USART data clock timing example (M=0)

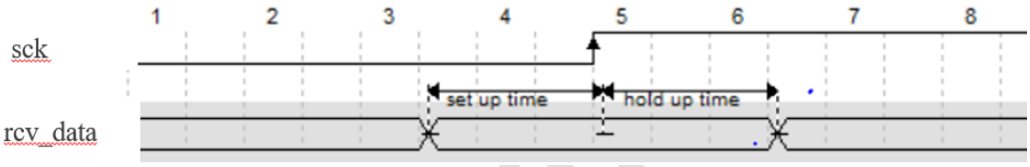


Figure 27-11 RX data sample/hold time (M=1)

27.2.11. Single-line half-duplex communication

Single-wire, half-dual mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must remain clear:

- LINEN and CLKEN bits of USART_CR2 registers
- SCEN and IREN bits of the USART_CR3 register

The USART can be configured to follow a single-wire half-duplex protocol. In single-wire half-duplex mode, the TX and RX pins are internally interconnected. Half-duplex and full-duplex communication is selected using the control bit "HALF DUPLEX SEL" (HDSEL bit in USART_CR3).

When HDSEL is "1":

- RX is no longer used
- TX is always released when no data is transmitted. Therefore, it behaves as a standard I/O port in the idle or receive state. This means that this I/O must be configured as a dangling input (or open-drain output high) when not driven by USART.

Other than this, communication is similar to normal USART mode. Conflicts on the line are managed by software (e.g., through the use of a central arbiter). In particular, transmissions are never blocked by hardware. When the TE bit is set, the transmission continues as soon as the data is written to the data register.

27.2.12. Smart card

Setting the SCEN bit of the USART_CR3 register selects smart card mode. In smart card mode, the lower column bits must be kept clear:

- LINEN bit of the USART_CR2 register
- HDSEL bit and IREN bit of the USART_CR3 register

In addition, the CLKEN bit can be set to provide a clock to the smart card.

The interface is ISO7816-3 compliant and supports smart card asynchronous protocols. The USART should be set to:

- 8 data bits plus parity bit: M=1, PCE=1 in USART_CR1 register.
- 1.5 stop bits for transmit and receive: STOP=11 in the USART_CR2 register.

Note: It is also possible to select 0.5 stop bits on receive, but to avoid switching between the 2 configurations, it is recommended to use 1.5 stop bits on transmit and receive.

The example given below illustrates the signaling on the data line with and without a checksum error.

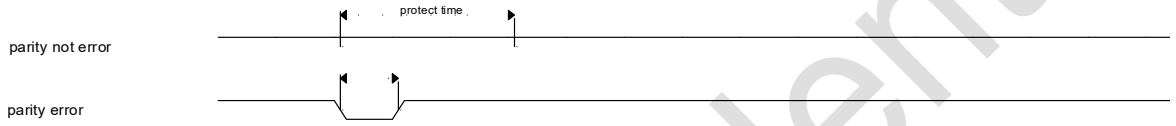


Figure 27-12 ISO 7816-3 Asynchronous Protocol

When connected to a smart card, the TX of the USART drives a bi-directional line that is also driven by the smart card. To do this, SW_RX must be connected to the same I/O port as TX. The transmitter's output enable bit TX_EN is set during the sending of the start bit and the data byte, and is released during the sending of the stop bit (weak pull-up), so that the receiver can pull the data line low if a checksum error is detected. If TX_EN is not used, TX is pulled high during the stop bit: in that case the receiver can also drive the line as long as TX is configured as open drain.

Smart card is a single-wire, half-duplex communication protocol

- Sending data out from the transmit shift register is delayed by a minimum of 1/2 baud clock. In normal operation, a full transmit shift register will begin shifting data out on the next baud clock edge. In smart card mode, this send is delayed by 1/2 baud clock.
- If a parity error is detected during the reception of a data frame set to 0.5 or 1.5 stop bits, the transmit line is pulled down one baud clock cycle after the reception of the frame is completed (i.e., at the end of the stop bits). This is to tell the smart card that the data sent to the USART was not received correctly. This NACK signal (pulling the transmit line down one baud clock cycle) will generate a frame error on the transmit side (the transmit side is configured for 1.5 stop bits). The application program can handle resending the data according to the protocol. If the NACK control bit is set, the receiver will give a NACK signal when a checksum error occurs; otherwise no NACK is sent.
- The setting of the TC flag can be delayed by programming the protection time register. In normal operation, the TC is raised when the Transmit Shift Register becomes empty and no new transmit requests are made. In smart card mode, an empty transmit shift register triggers the protection time counter to start counting up until the value in the protection time register is reached. TC is forced low during this time. When the protection time counter reaches the value in the protection time register, TC is set high.

- The revocation of the TC flag is not affected by the smart card mode.
- If the transmitter detects a frame error (receives a NACK signal from the receiver), the transmitter's receive function module does not detect the NACK as a start bit. The duration of the received NACK can be 1 or 2 baud clock cycles according to the ISO protocol.
- On the receiver side, if a checksum error is detected and a NACK is sent, the receiver does not detect the NACK as a start bit.

Note:

- The break symbol has no meaning in smart card mode. A 00h data with a framing error will be treated as data instead of a break symbol.
- No IDLE frames are sent when switching the TE bit back and forth. The ISO protocol does not define IDLE frames.

The following figure details how the USART samples the NACK signal. In this example, the USART is sending data and is configured for 1.5 stop bits. In order to check the integrity of the data and the NACK signal, the USART's receive function block is activated.

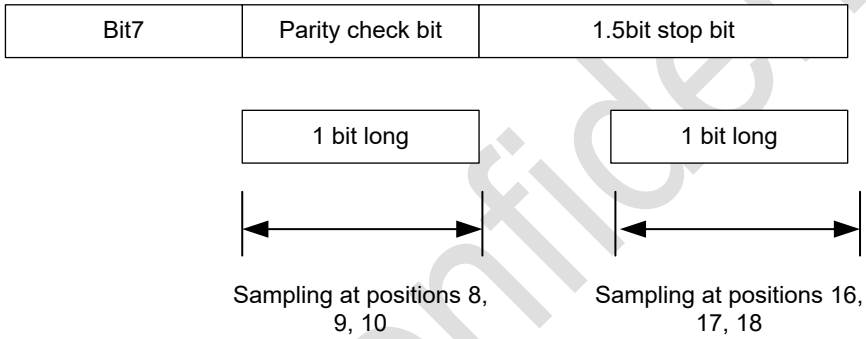


Figure 27-13 Use 1.5 stop bit to detect parity check errors

The USART can provide a clock to the smart card via the CK output. In smart card mode, CK is not directly linked to communication, but first simply drives the smart card clock with the internal peripheral input clock via a 5-bit prescaler. The division frequency is configured in the prescaler register USART_GTPR. The CK frequency can be from fCK/2 to fCK/62, where fCK is the peripheral input clock.

27.2.13. IrDA SIR ENDEC Function Module

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IRDA mode, the lower column bits must be kept clear:

- LINEN,STOP and CLKEN bits of the USART_CR2 registers
- SCEN and HDSEL bits of the USART_CR3 register.

27.2.13.1. IrDA Normal Mode

The IrDA SIR physical layer specifies the use of an inverted-zero modulation scheme (RZI), which uses an infrared light pulse to represent a logic '0' (see Figure 4-12). The SIR transmitter encoder modulates the NRZ (non-return-to-zero) bit stream output from the USART. The output pulse stream is transmitted to an external output driver and IR LED. The USART only supports up to 115.2 Kbps for the SIR ENDEC. In normal mode, the pulse width is specified as 3/16 of a bit period.

- The SIR receive decoder demodulates the nulled bit stream from the IR receiver and outputs

the received NRZ serial bit stream to the USART.In the idle state, the decoder input is normally high (marking state). The transmit encoder output has the opposite polarity of the decoder input. When the decoder input is low, a start bit is detected.

- IrDA is a half-duplex communication protocol. If the transmitter is busy (that is, USART is sending data to the IrDA encoder), any data on the IrDA receive line will be ignored by the IrDA decoder. If the receiver is busy (that is, the USART is receiving decoded data from the IrDA decoder), data on the TX from the USART to IrDA will not be encoded by IrDA. When receiving data, sending should be avoided because the data that will be sent may be corrupted.
- The SIR transmit logic sends '0' as a high pulse and '1' as a low level. The width of the pulse is specified as 3/16 of the bit period in normal mode (see Figure 4-13).
- The SIR receive logic interprets high level states as '1' and low pulses as '0'.
- The transmit encoder output has opposite polarity to the decoder input. When idle, the SIR output is in the low state.
- The SIR decoder converts the IrDA-compatible receive signal into a bitstream for the USART.
- The IrDA specification requires that the pulse be wider than 1.41us. The pulse width is programmable. The spike pulse detection logic on the receiver side filters out pulses that are less than 2 PSC cycles wide (PSC is a prescaled value programmed in the IrDA low power baud rate register USART_GTPR). Pulses less than 1 PSC cycle in width must be filtered out, but those with a width greater than 1 and less than 2 PSC cycles may be received or filtered out, and those with a width greater than 2 cycles will be considered a valid pulse. When PSC=0, the IrDA encoder/decoder does not operate.
- The receiver can communicate with a low power transmitter.
- In IrDA mode, the STOP bit on the USART_CR2 register must be configured as a stop bit.

27.2.13.2. IrDA Low Power Mode

Transmitter:

In low-power mode, the pulse width no longer lasts 3/16 of a bit period. Instead, the width of the pulse is 3 times the low-power baud rate, which can be as low as 1.42 MHz. Typically this value is 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). A low power mode programmable frequency divider divides the system clock to achieve this value.

Receiver:

Low power mode reception is similar to normal mode reception. To filter out spiky interference pulses, the USART should filter out pulses shorter than 1 PSC in width. Only low level signals from the IrDA low power baud rate clock (PSC in USART_GTPR) with a duration greater than 2 cycles are accepted as valid.

Notes:

1) Pulses with widths less than 2 greater than 1 PSC period may or may not be filtered.
2) The receiver build-up time should be managed by software. The IrDA Physical Layer Specification specifies a minimum delay of 10ms between transmission and reception (IrDA is a half-duplex protocol).

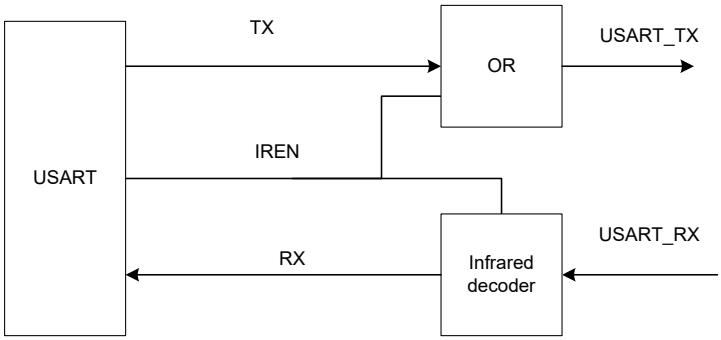


Figure 27-14 IrDA SIR ENDEC Block Diagram

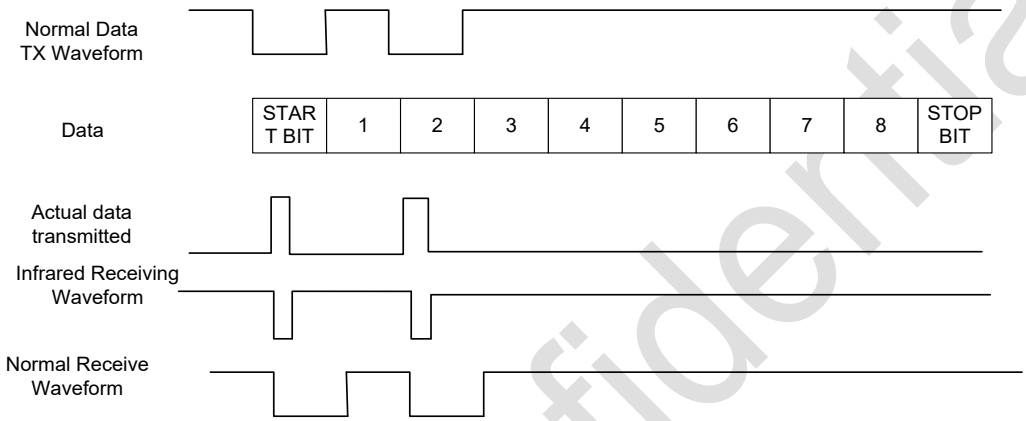


Figure 27-15 IrDA Data Modulation (3/16) - Normal Mode

27.2.14. Continuous communication using DMA

The USART can communicate continuously using DMA. DMA requests are generated separately for Rx buffers and Tx buffers.

Note: Refer to the product technical description to determine if a DMA controller is available. In the USART2_SR register, the TXE/RXNE flags can be cleared to
enable continuous communication.

Send using DMA:

Transmission using DMA can be activated by setting the DMAT bit on the USART_CR3 register. When the TXE bit is set to '1', the DMA transfers data from the specified SRAM area to the USART_DR register

- The memory address is configured on the DMA control register as the source address for DMA transfers. After each TXE event, data will be read from this memory area and transferred to the USART_DR register.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure the channel priority on the DMA register.
- Configure whether to generate a DMA interrupt when the transfer is half or fully completed, depending on the requirements of the application program.
- Activate the channel on the DMA register.

When the transfer of the amount of data specified by the DMA controller is complete, the DMA controller generates an interrupt on the interrupt vector of that DMA channel. In transmit mode,

when the DMA has finished transmitting all the data to be sent, the DMA controller sets the TCIF flag of the DMA_ISR register; monitoring the TC flag of the USART_SR register confirms the end of the USART communication, which avoids corrupting the last transmitted data before shutting down the USART or entering the shutdown mode; the software needs to wait for TXE = 1 and then wait for TC = 1.

Receive using DMA:

Receiving using DMA can be activated by setting the DMAR bit in the USART_CR3 register. Each time a byte is received, the DMA controller transfers the data from the USART_DR register to the specified SRAM area (refer to the DMA related description). To assign a DMA channel for USART reception, proceed as follows (x indicates the channel number):

- The USART_DR register address is configured as the source address for the transfer via the DMA control register. After each RXNE event, data will be read from this address and transferred to memory.
- Configure the memory address as the destination address for the transfer via the DMA control register. After each RXNE event, data will be transferred from USART_DR to this memory area.
- The total number of bytes to be transferred is configured in the DMA control register.
- Configure the channel priority on the DMA register.
- Configure whether to generate a DMA interrupt when the transfer is half or fully completed, depending on the requirements of the application program.
- Activate the channel on the DMA control register.

The DMA controller generates an interrupt on the interrupt vector of the DMA channel when reception of the amount of transmission specified by the DMA controller is completed.

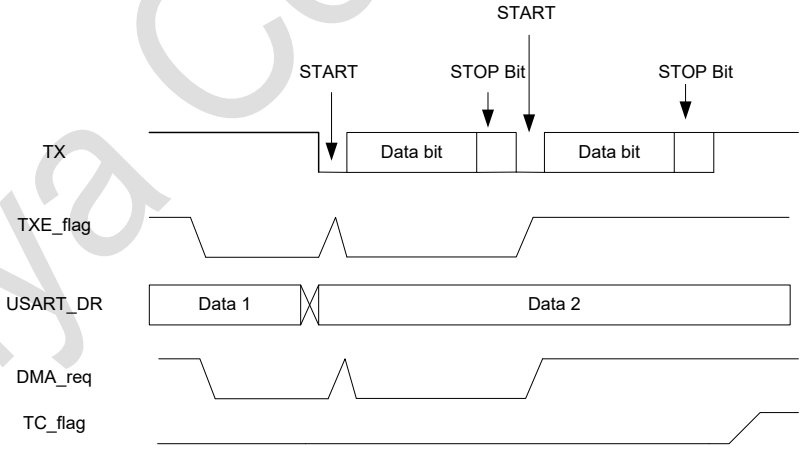


Figure 27-16 Using DMA to send

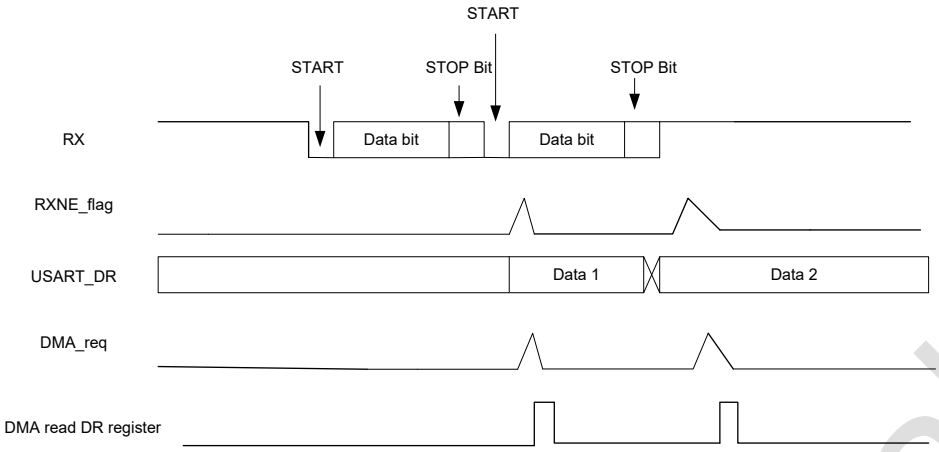


Figure 27-17 Using DMA to Receive

Error flag and interrupt generation in multi-buffer communication

In the case of multi-buffer communication, if any error occurs during communication, the error flag is raised after the current byte has been transmitted.

If the interrupt enable bit is set, an interrupt is generated. In the case of single byte reception, the frame error, overflow error, and noise flags, which are set together with RXNE, have separate error flag interrupt enable bits; if set, an interrupt is generated at the end of the current byte transmission.

27.2.15. Hardware flow control

The nCTS input and nRTS output can be used to control the serial data flow between two devices. The fol lowing diagram shows how to connect two devices in this mode.

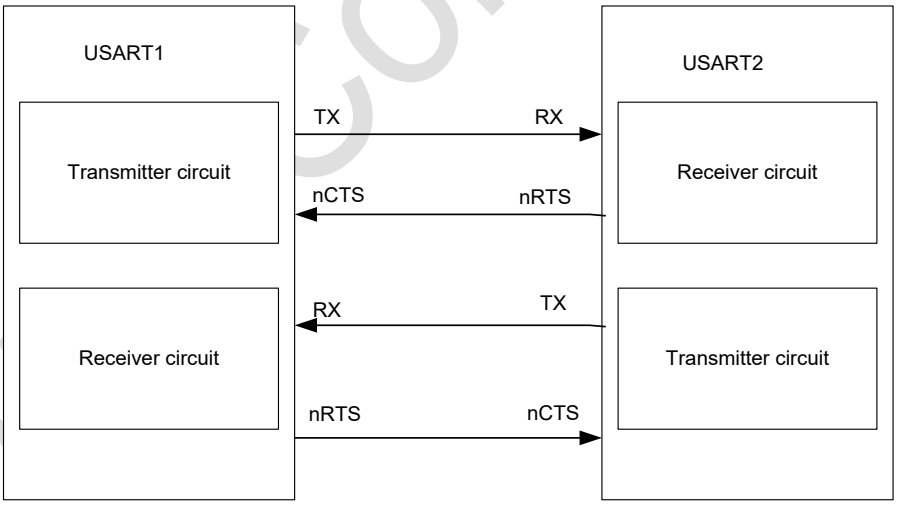


Figure 27-18 Hardware flow control between two USARTs

RTS and CTS flow control can be enabled independently by setting RTSE and CTSE in UASRT_CR3, respectively.

RTS flow control:

If RTS flow control is enabled (RTSE = 1), nRTS becomes active (connected low) whenever the USART receiver is ready to receive new data. When data arrives in the receive register, nRTS is released, thus indicating that it is desired to stop data transmission at the end of the current frame. The following figure shows an example of communication with RTS flow control enabled.

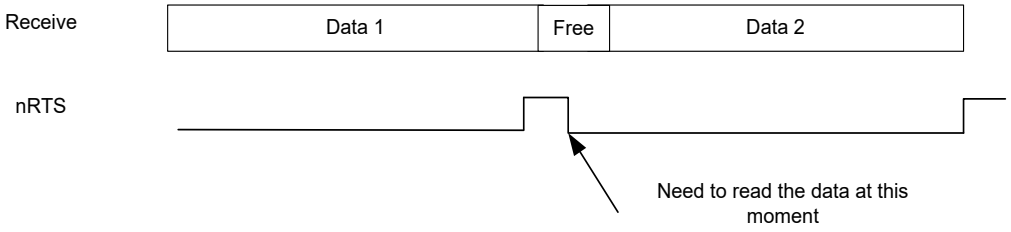


Figure 27-19 RTS flow control

CTS Flow Control

If CTS flow control is enabled (CTSE=1), the transmitter checks the nCTS input before sending the next frame. If nCTS is valid (pulled low), the next data is sent (assuming that that data is ready to be sent, i.e. TXE=0), otherwise the next frame is not sent. If nCTS is made invalid during transmission, sending stops when the current transmission is completed.

When CTSE=1, the hardware automatically sets the CTSIF status bit as soon as the nCTS input changes state. It indicates whether the receiver is ready to communicate. If the CTSIE bit of the USART_CT3 register is set, an interrupt is generated. The following figure shows an example of enabling CTS flow control communication.

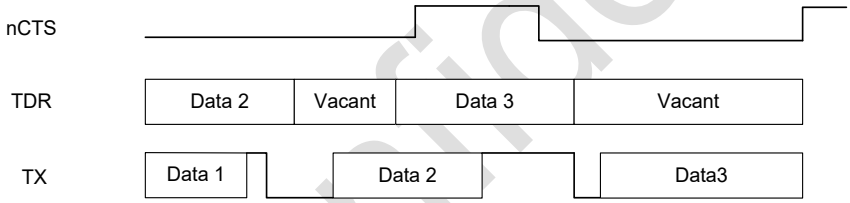


Figure 27-20 CTS flow control

27.2.16. Interrupt request

Table 3-5 USART Interrupt Request

No.	Interrupt event	Event Flag	Enable Bit	Transmit/Receive
1	Send Data Register Empty	TXE	TXEIE	Transmit
2	CTS (Clear to Send) interrupt	CTSIF	CTSIE	Transmit
3	Transmission complete	TC	TCIE	Transmit
4	Receive register not empty (read data ready)	RXNE	RXNEIE	Receive
5	Overrun error	ORE		Receive
6	Free frame	IDLE	IDLEIE	Receive
7	Parity check error	PE	PEIE	Receive
8	Disconnect Flag	LBD	LBDIE	Receive
9	Noise, overrun and frame errors in multiprocessor communication	NR/ORE/FE	EIE	Receive

This flag bit is used only when data is received using DMA.

The various interrupt events of the USART are connected to the same interrupt vector (see below) with the following various interrupt events:

- During transmit: transmit complete, clear transmit, transmit data register empty.

- During receive: idle bus detection, overflow error, receive data register non-empty, checksum error, LIN break symbol detection, noise flag (only in multi-buffer communication) and frame error (only in multi-buffer communication).

These events can generate their own interrupts if the corresponding enable control bits are set.

The circuit is implemented as:

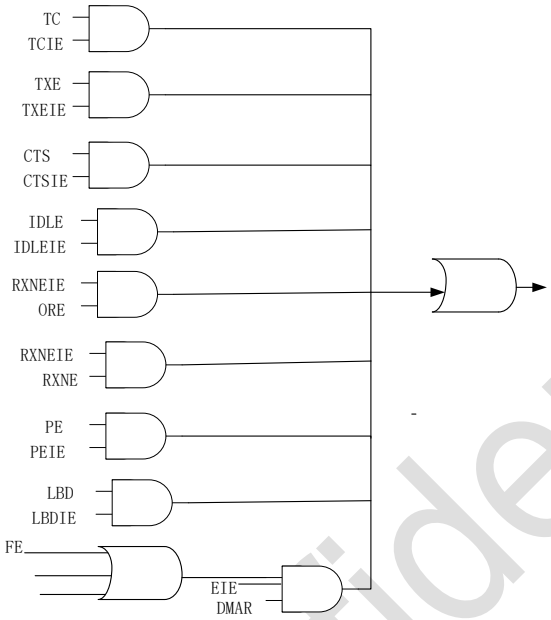


Table 27-4 USART Interrupt Image Map

27.3. Register description

27.3.1. Control Register (SR)

Address Offset: 0x00

Reset Value: 0x00c0

12	11	10	9	8	7	6	5	4	3	2	1	0
ABRRQ	ABRE	ABRF	CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
W	R	R	Rc_W0	Rc_W0	R	RC_W0	RC_W0	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:10	Reserved	RES	\	\
12	ABRRQ	W	0	Auto Baud Rate Request. A write of 1 to this bit resets the ABRF flag bit and requests automatic baud rate detection for the next frame.
11	ABRE	R	0	Auto Baud Rate Error Flag. Hardware sets this register when there is an error in the automatic baud rate detection (baud rate out of range or character comparison error). Software clears this bit by writing a 1 to the ABRRQ register.
10	ABRF	R	0	Auto baud rate detection flag.

Bit	Name	R/W	Reset Value	Function
				This bit is set by hardware when the auto baud rate is set (also set RXNE=1 to generate an interrupt when the interrupt is enabled), or when there is an error in the auto baud rate detection operation (ABRE=1, RXNE=1, FE=1). Software clears this bit by writing a 1 to the ABRRQ bit of the USART_RQR register.
9	CTS	Rc_W0	0	CTS: CTS Flag If the CTSE bit is set, this bit is set high by hardware when the nCTS input changes state. It is cleared by software. If the CTSIE in USART_CR3 is '1', an interrupt is generated. 0: No change on the nCTS status line; 1: A change occurs on the nCTS status line.
8	LBD	Rc_W0	0	LBD: LIN break detection flag When a LIN break is detected, this bit is set '1' by hardware and cleared '0' by software (write 0 to this bit). If LBDIE = 1 in USART_CR3, an interrupt is generated. 0: No LIN disconnect detected; 1: LIN disconnect detected. Note: If LBDIE = 1, an interrupt is to be generated when LBD is '1'
7	TXE	R	1	TXE: Transmit data register empty This bit is set by hardware when the data in the TDR register is transferred to the shift register by hardware. If USART_CR1 register is 1, an interrupt is generated. A write operation to USART_DR clears this bit to zero. 0: Data has not been transferred to the shift register; 1: Data has been transferred to the shift register. Note: This bit is used in single buffer transfers.
6	TC	RC_W0	1	Transmission complete When the transmission of a frame containing data is complete and TXE=1, the hardware sets this bit to 1. If TXEIE in the USART_CR1 register is 1, an interrupt is generated. A write operation to USART_DR clears this bit to zero. 0: Transmission is not yet complete; 1: the transmission is completed.
5	RXNE	RC_W0	0	Read data register not empty When the data in the RDR shift register is transferred to the USART_DR register, this bit is set by hardware. If RXNEIE in the USART_CR1 register is 1, an interrupt is generated. A read operation of USART_DR clears this bit to zero. the RXNE bit can also be cleared by writing a 0. This clearing procedure is recommended only for multi-buffer communications. 0: Data is not received

Bit	Name	R/W	Reset Value	Function
				1: Data received and can be read
4	IDLE	R	0	IDLE: Bus idle detected This bit is set by hardware when bus idle is monitored. If IDLEIE in USART_CR1 is 1, an interrupt is generated. This bit is cleared due to a software sequence (read USART_SR first; then USART_DR). 0: No idle bus detected; 1: Idle bus detected Note: The IDLE bit will not be set high again until the RXNE bit is set (i.e., another idle bus is detected)
3	ORE	R	0	ORE (overrun error) When RXNE=1, the hardware bits the data currently being received in the shift register that needs to be transferred to the RDR register. An interrupt is generated if RXNEIE in USART_CR1. It is cleared by a software sequence (read USART_SR first, then USART_CR). 0: No overload error; 1: overload error detected. Note: When this bit is set, the value in the RDR register is not lost, but the data in the shift register is overwritten. If the EIE bit is set, in multi-buffer communication mode, the ORE flag set will generate an interrupt of.
2	NE	R	0	NE: Noise error flag This bit is set by hardware when noise is detected in a received frame. It is cleared to zero by the software sequence (first USART_SR, then read USART_DR). 0: No noise detected; 1: noise detected. Note: This bit does not generate an interrupt because it appears with RXNE and hardware generates an interrupt when the RXNE flag is set. In multi-buffer communication mode, if the EIE bit is set, an interrupt is generated when the NE flag is set
1	FE	R	0	FE: Framing error. This bit is set by hardware when a synchronization mismatch, excessive noise, or a break character is detected. It is cleared by a software sequence (USART_SR is read first, USART_DR is read later). 0: No frame error detected; 1: frame error or break character detected. Note: This bit does not generate an interrupt because it comes with RXNE and the hardware generates an interrupt when the RXNE flag is set.

Bit	Name	R/W	Reset Value	Function
				If the currently transmitted data generates both a frame error and an overload error, the hardware will still continue the transmission of that data and only set the ORE flag bit. In multi-buffer communication mode, if the EIE bit is set, an interrupt is generated when the FE flag is set.
0	PE	R	0	PE:Parity error In receive mode, if a parity error occurs, the hardware bit for this location. It is cleared by the software sequence (reading USART_SR and USART_DR in sequence). Before clearing the PE bit, software must wait for the RXNE flag bit to be pointed to 1. If PEIE in USART_CR1 is 1, an interrupt is generated. 0: No parity error; 1: parity error.

27.3.2. Data register (USART_DR)

Address Offset: 0x04

Reset Value: 0xxxxx

31:9	8	7	6	5	4	3	2	1	0
	DR[8:0]								
	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:9	Reserve	RES	\	\
8:0	DR[8:0]	RW	Undefined	Receive/Transmit Data Register. It consists of two registers in total (one TDR for transmit and one RDR for receive), so the DR register implements both read and write functions. The TDR register provides the parallel interface between the internal bus and the output shift registers (see Figure 248).The RDR register provides the parallel interface between the input shift registers and the internal bus. When the enable parity bit (the PCE position bit in USART_CR1) is transmitted, the value written to the MSB (which is either bit 7 or bit 8 depending on the length of the data) is replaced by a later parity bit.

27.3.3. Baud Rate Register (USART_BRR)

Note: The baud counter stops counting if TE or RE is disabled, respectively.

Address Offset: 0x08

Reset Value: 0x0000

31:16	15:4	3:0
Reserved bit	DIV_Mantissa[11:0]	DIV_Fraction[3:0]
RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:16	Reserved	RES	\	Reserved
15:4	DIV_Mantissa[15:4]	RW	0	12bit integer
3:0	DIV_Fraction[3:0]	RW	0	4bit decimal

27.3.4. Control Register (USART_CR)

27.3.4.1. Control Register 1 (USART_CR1)

Address Offset: 0x0C

Reset Value: 0x0000_0000

31:14													
Res													
13	12	11	10	9	8	7	6	5	4	3	2	1	0
UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:14	Reserved	RES	\	Reserved
13	UE	RW	0	UE:USART enable When this bit is cleared, the USART divider and outputs stop working after the current byte transfer is complete, thus reducing power consumption. This bit is set and cleared by software. 0: USART divider and outputs are disabled; 1: The USART module is enabled.
12	M	RW	0	M: Word length This bit defines the length of the data word, which is set and cleared by software. 0: one start bit, 8 data bits, n stop bits; 1: one start bit, 9 data bits, n stop bits. This bit cannot be modified during data transmission.
11	WAKE	RW	0	Wake: Wakeup method This bit determines the method of waking up the USART, and is set and cleared by software. 0: Wake up by idle bus; 1: Wakeup by address tag.
10	PCE	RW	0	PCE: Parity control enable Use this bit to select whether to perform hardware parity control (parity bit generation for transmit; parity bit detection for receive). When this bit is enabled, a parity bit is inserted

Bit	Name	R/W	Reset Value	Function
				in the highest bit of the transmitted data (if M=1, the highest bit is the 9th bit; if M=0, the highest bit is the 8th bit); the received data is checked for the parity bit. The software sets it to "1" or clears it to "0". Once this bit is set, the parity control will not take effect until the current byte transmission is completed. 0: Disable parity control; 1: Enable parity control.
9	PS	RW	0	PS: Parity selection When the parity control is enabled, this bit is used to select whether to use even parity or odd parity. The software sets it to "1" or clears it to "0". The selection takes effect after the current byte transmission is completed. 0: Even parity; 1: Odd parity.
8	PEIE	RW	0	PEIE: PE interrupt enable Set and cleared by software. 0: disable interrupt generation 1: PE interrupt enable
7	TXEIE	RW	0	TXE: TXE interrupt enable Software set and clear. 0: Disable; 1: TXE interrupt enable
6	TCIE	RW	0	TCIE: Transmit Completion Interrupt Enable This bit is set and cleared by software. 0:Disable interrupt generation. 1:Generate USART interrupt when ORE or RXNE in USART_SR is 1.
5	RXNEIE	RW	0	RXNEIE: RXNE interrupt enable This bit is set or cleared by software. 0: Interrupt generation is disabled 1: When ORE or RXNE in USART_SR is "1", USART interrupt is generated.
4	IDLEIE	RW	0	IDLE:IDLE interrupt enable This bit is set or cleared by software 0: Interrupt generation is disabled; 1: When IDLE in USART_SR is "1", USART interrupt is generated.
3	TE	RW	0	TE:Transmitter enable This bit enables the transmitter. This bit is set or cleared by software

Bit	Name	R/W	Reset Value	Function
				0: Disable transmission; 1: enables transmission. During data transmission, except in smart card mode, if there is a 0 pulse on the TE bit (i.e., after setting it to '0', then setting it to '1'), a "leading character" (idle bus) will be sent after the transmission of the current data word is completed. (idle bus) after the current data word has been transmitted. When TE is set, there is a delay of one bit time before the actual transmission starts.
2	RE	RW	0	RE: Receiver enable 0: Receive is disabled; 1: enables reception and starts searching for the start bit of the RX pin.
1	RWU	RW	0	RWU:Receiver wakeup This bit is used to determine whether or not to place the USART in silent mode. This bit is set or cleared by software. It is also cleared by hardware when the wakeup sequence arrives. 0: The receiver is in normal operating mode; 1: the receiver is in silent mode. Before placing the USART in silent mode (setting the RWU bit), the USART has first received a data byte. Otherwise, it cannot be woken up by idle bus detection in silent mode. When configured for Wake on Address Mark Detect (WAKE bit = 1), the RWU bit cannot be modified by software while the RXNE bit is set.
0	SBK	RW	0	SBK: Send break Software sets this register and sends the break byte. hardware clears this register after the stop bit of the break frame is sent. 0: No break byte is sent; 1: send break byte.

27.3.4.2. Control Register 2 (USART_CR2)

Address Offset: 0x10

Reset Value: 0x0000_0000

14	13	12	11	10	9	8	7	6	5	4	[3 : 0]
LINEN	STOP	STOP[1:0]	CLKEN	CPOL	CPHA	LBCL	保留	LBDIE	LBDL	保留	ADD[3:0]
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	Reserved	RES	-	Reserved
14	LINEN	RW	0	LIN mode enable. Software set and clear. 0: LIN mode; 1: LIN mode enable; LIN mode sends LIN synchronization breaks (13 low) and detects Lin synchronization breakers by enabling the SBK bit.
13: 12	STOP[1:0]	RW	2'b0	Stop bit configuration. 00: 1 stop bit; 01: 0.5 stop; 10: 2 stop bit; 11: 1.5 stop;
11	CLKEN	RW	0	Clock enable, this bit is used to enable the CK pin. 0: CK pin disabled; 1: CK pin enable; This bit is reserved when synchronization mode is not supported.
10	CPOL	RW	0	Clock Polarity. In synchronous mode, this bit is used to select the polarity of the clock output on the SLCK pin. It works in conjunction with the CPHA bit to produce the desired clock/data sampling relationship 0: Hold low on the CK pin when the bus is idle; 1: Hold high on the CK pin when the bus is idle.
9	CPHA	RW	0	Clock Phase In synchronous mode, this bit can be used to select the phase of the clock output on the SLCK pin. In conjunction with the CPOL bit to produce the desired clock/data sampling relationship 0: Data capture on the first edge of the clock; 1: data capture on the second edge of the clock.
8	LBCL	RW	0	Last bit clock pulse In synchronous mode, use this bit to control whether the clock pulse corresponding to the last byte (MSB) sent is output on the CK pin. 0: The clock pulse for the last bit of data is not output at the CK pin; 1: the clock pulse for the last bit of data is output at the CK pin.

Bit	Name	R/W	Reset Value	Function
				Note: The last data bit is the eighth or ninth bit sent (determined by the M bit in USART_CR1).
7	Reserved	RES	-	Reserved
6	LBDIE	RW	0	LIN break interrupt enable. 0: disable; 1: interrupt generation; This controls LBD in the USART_SR register so that LBD is 1 and an interrupt is generated.
5	LBDL	RW	0	LIN break detection length. 0: 10bits detection break; 1: 11bits detection break;
4	Reserved	RES	-	Reserved
3: 0	ADD[3:0]	RW	4'b0	USART Address. This register is used in multiprocessor silent mode and is used as the address when the 4bit address wakes up.

27.3.4.3. Control Register 3 (USART_CR3)

Address Offset: 0x14

Reset Value: 0x0000_0000

14:13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRMOD	ABREN	OVER8	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31：15	Reserved	RES		Reserved
14:13	ABRMOD[1:0]	RW	2'b0	Automatic baud rate detection mode. 00: Baud rate measurement from start bit; 01: falling edge to falling edge measurement; When ABREN=0 or UE=0, this register is write only. Note: This chip only supports 00 and 01 modes.
12	ABREN	RW	0	Auto baud rate enable. 0: Disable; 1: auto baud rate enable;
11	OVER8	RW	0	Oversampling mode. 0: Oversampling by 16; 1: Oversampling by 8;

Bit	Name	R/W	Reset Value	Function
				Note: This bit can only be written when UE = 0.
10	CTSIE	RW	0	CTS interrupt enable. 0: Disable; 1: CTS is 1 interrupt enable;
9	CTSE	RW	0	CTS enable. 0: CTS hardware flow control disabled; 1: CTS mode enable. Data is transmitted when there is when the CTS input is 0. In this case, when data is written to the data register, wait for CTS to be valid before initiating transmission.
8	RTSE	RW	0	RTS enable. 0: RTS hardware flow control disabled; 1: RTS output enable, the next data is requested only when the receive buffer is not full. The send operation is suspended after the current data is sent. If it is ready to receive data, set RTS to active (0).
7	DMAT	RW	0	DMA Enable Transmit. 0: Disable; 1: DMA enable on transmit;
6	DMAR	RW	0	DMA enable receive. 0: Disable; 1: DMA enable on receive;
5	SCEN	RW	0	Smart card mode enable. 0: disable; 1: enable;
4	NACK	RW	0	Smart card NACK enable. 0: Send NACK disable on parity error; 1: NACK enable is sent in case of parity error;
3	HDSEL	RW	0	Half-duplex selection. 0: Non-half-duplex mode; 1: Half-duplex mode selection;
2	IRLP	RW	0	IrDA low power consumption. 0: NORMAL mode; 1: IrDA low power mode;
1	IREN	RW	0	IrDA mode enable. Software enables and clears this register. 0: IrDA disable; 1: IrDA enable;
0	EIE	RW	0	Error interrupt enable.

Bit	Name	R/W	Reset Value	Function
				0: disable; 1: frame error FE, overrun error ORE, noise NF interrupt enable;

27.3.5. Guard Time and Preshunt Register (USART_GTPR)

Address Offset: 0x18

Reset Value: 0x0000_0000

[31:16]	[15:8]	[7:0]
Reserve	GT	PSC
RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	RES	-	Reserved
15: 8	GT[7:0]	RW	0	Guard time value. This field defines the protection time in baud clock units. This is required in smart card mode. The transmit complete flag is set when the protection time has elapsed.
7: 0	PSC[7:0]	RW	0	Prescaler value: 1. In infrared (IrDA) low-power mode: PSC[7:0] = IR low power baud rate. The system clock is divided to obtain the frequency in low power mode: The source clock is divided by the value in the register (only 8 bits are valid) 00000000: Reserved - do not write to this value; 00000001: divides the frequency of source clock 1; 00000010: divides the frequency to source clock 2; …… 2. In the normal mode of Infrared (IrDA): PSC can only be set to 00000001. 3. In smart card mode: PSC[4:0]: pre-scaler value The system clock is divided to provide a clock to the smart card. The value given in the register (lower 5 bits are valid) multiplied by 2 is used as the frequency division factor to the source clock. 00000: Reserved - do not write this value;

Bit	Name	R/W	Reset Value	Function
				00001: 2-division frequency to the source clock; 00010: 4-division frequency to the source clock; 00011: 6-division frequency to the source clock; ..... Note: Bits [7:5] have no meaning in smart card mode.

27.3.6. USART register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABRRQ	ABRE	ABRF	CTS	LBD.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value																				0	0	0	0		1	1	0	0	0	0	0	0
0x04	USART_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x08	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIV_Mantissa[11:0]												DIV_Faction[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1	USART_	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	L	S	R	C	C	C	L	R	L	L	R	ADD[3:0]			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CR2																		LBDIE								LBCL							
	Reset value																		0	0		0	0	0	0		0	0		0	0	0	0
0x14	USART_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABRMOD[1:0]		ABREN	OVER8	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN.	NACK	HDSEL	IRLP	IREN	EIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	Reserved	Res.																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

28. Clock Calibration Controller (CTC)

28.1. Introduction

The Clock Calibration Controller (CTC) uses hardware to automatically calibrate the internal 48MHz RC crystal (HSI48M). When the USBD module uses the HSI48M clock as the clock source, the HSI48M clock frequency must be required to be within the range of 48MHz±500ppm, but the internal crystal without calibration cannot meet such a high level of accuracy. the CTC module calibrates the HSI48M clock frequency based on the external high-precision reference source, and adjusts the calibration value automatically or manually to obtain an accurate HSI48M clock.

28.1.1. Main features

The CTC module accomplishes the following functions:

- Three external reference signal sources: GPIO, LSE clock, USBD_SOF
- Provides software reference synchronization pulses;
- Hardware auto-calibration, no software operation required;
- 16 bits calibration counter with reference signal source capture and reload functions;
- 8 bits clock calibration base for frequency evaluation and auto-calibration;
- Flag bits and interrupts to indicate the status of the clock calibration: calibration success status (CKOKIF), warning status (CKWARNIF) and error status (ERRIF).

28.1.2. Module block diagram

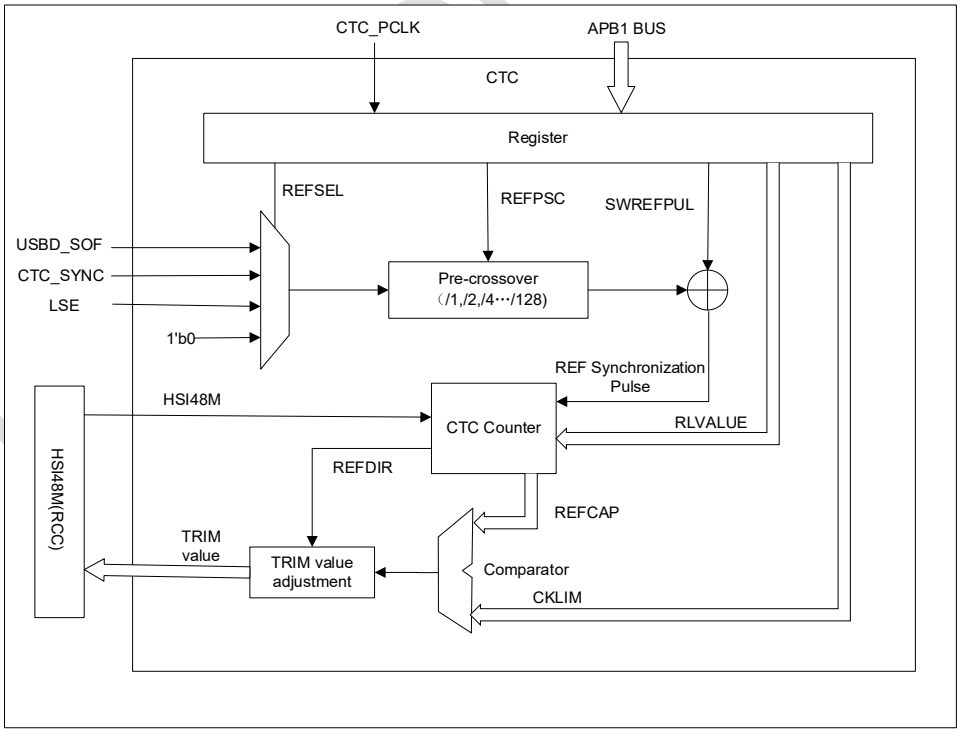


Figure 28-1 CTC module block diagram

28.2. Functional description

28.2.1. REF Synchronized Pulse Generator

First, the reference signal source is selected by setting the REFSEL bit in the CTC_CTL1 register (CTC control register 1): GPIO, LSE clock output or USBD_SOF.

Then, you can configure the signal polarity when the reference signal source is synchronized by setting the REFPOL bit in the CTC_CTL1 register, and generate an appropriate synchronized clock frequency signal by setting the REFPSC bit in the CTC_CTL1 register.

If a software reference pulse signal is to be used, the SWREFPUL bit in the CTC_CTL0 register (CTC control register 0) needs to be set to 1. The software reference pulse signal and the external reference pulse signal are finally subjected to a logic 'or' operation.

28.2.2. CTC Calibration Counter

The CTC clock calibration counter is clocked by the HSI48M. After setting the CNTEN bit in the CTC_CTL0 register, when the first REF synchronization pulse signal is detected, the counter starts counting down from the RLVALUE value (RLVALUE is defined in the CTC_CTL1 register). Each time a REF synchronization pulse signal is detected, the counter reloads the RLVALUE value and starts counting down again. If a REF synchronization pulse signal is never detected, the counter counts down to zero, then counts up to 128 x CKLIM (CKLIM is defined in CTC_CTL1), and finally stops until the next REF synchronization pulse signal is detected. Once the REF synchronization pulse signal is detected, the count value of the current CTC calibration counter is captured and deposited into the REFCAP bit in CTC_STAT CTC Status Register), while the count direction of the current counter is deposited into the REFDIR bit in CTC_STAT. The details are shown below.

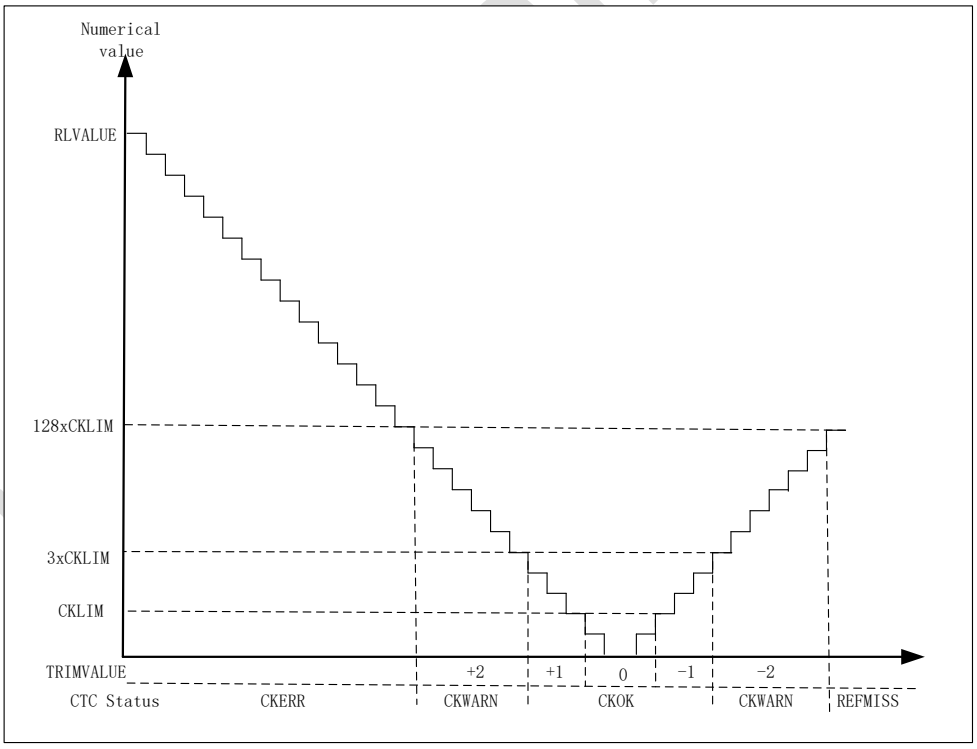


Figure 28-2 CTC Calibration Counter

28.2.3. Frequency evaluation and automatic calibration process

The clock frequency evaluation function starts to execute when the REF synchronization pulse signal appears. If the REF synchronization pulse signal appears during the counter down count, it means that the current clock frequency is slower than the desired clock frequency (frequency of 48M), and it is necessary to increase the TRIMVALUE value (clock calibration value) in CTC_CTL0. If the REF

synchronization pulse signal appears in the process of counter counting up, it means that the current clock frequency is faster than the desired clock frequency, and it is necessary to decrease the TRIMVALUE value. The CKOKIF bit, the CKWARNIF bit, the CKERR bit, and the REFMISS bit in CTC_STAT reflect the status of the frequency evaluation.

The hardware auto-calibration mode is enabled if the AUTOTRIM (hardware auto-calibration mode) position 1 in CTC_CTL0. In this mode, if the REF synchronization pulse signal appears during the counter down count, it means that the current clock frequency is slower than the desired clock frequency, and the TRIMVALUE value in CTC_CTL0 will be increased automatically to increase the current clock frequency. On the other hand, if the REF synchronization pulse signal appears in the process of counter counting up, it means that the current clock frequency is faster than the desired clock frequency, and the TRIMVALUE value will be decreased automatically to reduce the current clock frequency.

- Counter < CKLIM, REF synchronization pulse signal detected:
  - The CKOKIF bit (Clock Calibration Success Flag Bit) in CTC_STAT is set, and at the same time, if the CKOKIE bit (Clock Calibration Completed Interrupt Enable Bit) in CTC_CTL0 is set to 1, an interrupt will be generated. If AUTOTRIM in CTC_CTL0 is set to 1, the TRIMVALUE value in CTC_CTL0 remains unchanged.
- CKLIM ≤ Counter < 3 x CKLIM, REF synchronization pulse signal detected:
  - The CKOKIF bit in CTC_STAT is set, and an interrupt will be generated if the CKOKIE position in CTC_CTL0 is 1. If the AUTOTRIM bit in CTC_CTL0 is set to 1, the TRIMVALUE value in CTC_CTL0 will be incremented by 1 during the downward counter count and decremented by 1 during the upward counter count.
- 3 x CKLIM ≤ Counter < 128 x CKLIM, REF synchronization pulse signal detected:
  - The CKWARNIF bit (Clock Calibration Warning Interrupt Bit) in CTC_STAT is set, and at the same time, if the CKWARNIE bit (Clock Calibration Warning Interrupt Enable Bit) in CTC_CTL0 is set to 1, an interrupt will be generated. If the AUTOTRIM bit in CTC_CTL0 is set to 1, the TRIMVALUE value in CTC_CTL0 will be incremented by 2 during the downward counter count and decremented by 2 during the upward count.
- Counter ≥ 128 x CKLIM, the counter detects the REF synchronization pulse signal during down count:
  - The CKERR bit (clock calibration error bit) in CTC_STAT is set, and an interrupt will be generated if the ERRIE bit (error interrupt enable bit) in CTC_CTL0 is set to 1. The TRIMVALUE value in CTC_CTL0 remains unchanged.
- Counter = 128 x CKLIM, the counter is in the process of counting up:
  - The REFMISS bit (REF Synchronization Pulse Loss bit) in CTC_STAT is set, and an interrupt will be generated if the ERRIE bit in CTC_CTL0 is set to 1. The TRIMVALUE value in CTC_CTL0 remains unchanged.

If the calibration value of TRIMVALUE in CTC_CTL0 is greater than 127, an overflow event will occur, and at the same time, if the calibration value of TRIMVALUE is less than 0, an underflow

event will occur. The value of TRIMVALUE ranges from 0 to 127 (the value of TRIMVALUE is 127 for an overflow event and 0 for an underflow event). The TRIMERR bit (calibration value error bit) in CTC_STAT will then be set and an interrupt will be generated if the ERRIE bit in CTC_CTL0 is set to one.

28.2.4. Software programming guide

The RLVALUE bit and the CKLIM bit in CTC_CTL1 are key for clock frequency evaluation and hardware auto-calibration. Their values are calculated from the frequency of the desired clock (HSI48M: 48 MHz) and the frequency of the REF synchronization pulse signal. Ideally, the REF synchronization pulse signal occurs when the CTC counter counts to zero, so the value of RLVALUE is:

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

The value of CKLIM is set by the user according to the accuracy of the clock, and it is generally recommended to set it to half of the step size, so the value of CKLIM is:

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

Typical step value is 0.12%, F_{clock} is the frequency of the desired clock (48MHz), and F_{REF} is the frequency of the REF synchronization pulse signal.

TRIMVALUE in CTC_CTL0 can be written by software when the AUTOTRIM bit is 0. However, modifying TRIMVALUE will directly affect the frequency of the HSI-48M clock, so TRIMVALUE should not be modified by software at will. it is recommended that the user modifies it in the middle of the two reference signals based on the judgment of the flag bit; or if there is already a reliable value. Users can directly modify the value of HSI48TRIM in RCC_CFGR1.

28.3. Register description (base address 0x4000_C800)

28.3.1. Control register 0 (CTC_CTL0)

Address offset: 0x00

Reset value: 0x0000 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TRIMVALUE[6:0]							SWREFPUL	AUTOTRIM	CNTEN	Res.	EREFIE	ERRIE	CKWARNIE	CKOKIE
	RW	RW	RW	RW	RW	RW	RW	W	RW	RW		RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31:15	Reserved	RES	-	Reserved
14:8	TRIMVALUE[6:0]	RW	7’b1000000	HSI48M Calibration values .

Bit	Name	R/W	Reset Value	Function
				When the AUTOTRIM value in CTC_CTL0 is 0, this bit is set and cleared by software, this mode is used for software calibration process. When the AUTOTRIM value in CTC_CTL0 is 1, this bit is read-only and is automatically modified by hardware; this mode is used for the hardware calibration process. The middle value of TRIMVALUE is 64. When the TRIMVALUE value is increased by 1, the HSI48M clock frequency is increased by approximately 57 KHz. When the TRIMVALUE value is decreased by 1, the HSI48M clock frequency is decreased by approximately 57 KHz.
7	SWREFPUL	W	0	Software generates a synchronization reference signal pulse. This bit is set by software and provides a synchronization reference pulse signal for the CTC counter. This bit is automatically cleared by hardware and returns 0 for read operations. 0: no effect; 1: Software generates a synchronization reference pulse signal;
6	AUTOTRIM	RW	0	Hardware auto-calibration mode. This bit is set or cleared by software. When this bit is 1, the hardware auto-calibration mode is enabled, and the TRIMVALUE value in CTC_CTL0 is continuously and automatically modified by hardware until the clock frequency of the HSI48M reaches 48MHz. 0: Disable hardware auto-calibration mode 1: Enable hardware auto-calibration mode
5	CNTEN	RW	0	CTC Counter enable. This bit is set or cleared by software to enable or disable the CTC counter. When this bit is 1, the value of CTC_CTL1 cannot be modified. 0: Disable CTC counter 1: Enable CTC counter
4	Reserved	RES	-	Reserved
3	EREFIE	RW	0	Expected reference signal interrupt enable. 0: Expected reference signal interrupt disabled 1: Enable interrupt generation from desired reference signal
2	ERRIE	RW	0	Error interrupt enable. 0: Disable error interrupt 1: Enable error interrupt
1	CKWARNIE	RW	0	Clock calibration warning interrupt enable. 0: Disable clock calibration warning interrupt 1: Enable clock calibration warning interrupt

Bit	Name	R/W	Reset Value	Function
0	CKOKIE	RW	0	Clock calibration completion interrupt enable. 0: disable clock calibration completion interrupt 1: Enable clock calibration completion interrupt

28.3.2. Control register 1 (CTC_CTL1)

Address offset: 0x04

Reset value: 0x2022 BB7F

This register can only be accessed by word (32 bits). When CNTEN is 1, the value of this register cannot be modified.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFPOL	Res.	REFSEL[1:0]		Res.	REFPSC[2:0]			CKLIM[7:0]							
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
RLVALUE[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	REFPOL	RW	0	Reference Signal Source Polarity. This bit is set or cleared by software to select the synchronization polarity of the reference signal source. 0: Select rising edge 1: Select falling edge
30	Reserved	RES	-	Reserved
29:28	REFSEL[1:0]	RW	2'b10	Reference Signal Source Selection. This bit is set or cleared by software to select the reference signal source. 00: Select GPIO input signal 01: Select LSE clock 10: Select USBD_SOF signal 11: Reserved, select 0
27	Reserved	RES	-	Reserved
26:24	REFPSC[2:0]	RW	3'b000	Reference signal source prescaler. This bit is set or cleared by software. 000: Reference signal no frequency division 001: Reference signal 2-division frequency; 010: Reference signal 4-division frequency; 011: Reference signal 8-division frequency; 100: Reference signal 16-division frequency;

Bit	Name	R/W	Reset Value	Function
				101: Reference signal 32-division frequency; 110: Reference signal 64-division frequency; 111: Reference Signal 128-division frequency
23:16	CKLIM[7:0]	RW	0x22	Clock calibration time base limit. This bit is set or cleared by software to define the clock calibration time base limit. This bit is used in the frequency evaluation and auto-calibration process.
15:0	RLVALUE[15:0]	RW	0xBB7F	CTC counter reload value. This bit is set or cleared by software to define the CTC counter reload value that will be reloaded into the CTC calibration counter when a synchronization reference pulse is detected.

28.3.3. Status register (CTC_SR)

Address offset:0x08

Reset value:0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFCAP[15:0]															
R	R	R	R	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
REFDIR	Res				TRIMERR	REFMISS	CKERR	Res				EREFIF	ERRIF	CKWARNIF	CKOKIF
R					R	R	R					R	R	R	R

Bit	Name	R/W	Reset Value	Function
31:16	REFCAP[15:0]	R	0x0000	The CTC counter captures the value. When a synchronized reference pulse signal is detected, the count value in the CTC calibration counter is deposited into the REFCAP bit.
15	REFDIR	R	1	CTC calibration clock count direction. When a synchronized reference pulse signal is detected, the count direction of the CTC calibration counter is stored in the REFDIR bit. 0: Count up 1: Count down
14:11	Reserved	RES	-	Reserved
10	TRIMERR	R	0	Calibration value error bit.

Bit	Name	R/W	Reset Value	Function
				This bit is set by hardware when there is an overflow or underflow of the TRIMVALUE value in CTC_CTL0. If ERRIE in CTC_CTL0 is in position 1, an interrupt is generated. The TRIMERR bit can be cleared to zero by writing 1 to the ERRIC bit in CTC_INTC. 0: No calibration value error occurred. 1: Calibration value error occurred.
9	REFMISS	R	0	Synchronization reference pulse signal loss. This bit is set by hardware when the synchronization reference pulse signal is lost. REFMISS bit is set when the CTC calibration counter does not detect a sync reference pulse signal for 128 x CKLIM counts in incremental counting. Indicates that the current clock is too fast to calibrate to the desired frequency value, or some other error has been generated. The REFMISS bit can be cleared to zero by writing a 1 to the ERRIC bit in CTC_INTC. 0: No synchronization reference pulse signal loss. 1: Synchronized reference pulse signal lost.
8	CKERR	R	0	Clock calibration error bit. This bit is set by hardware when a clock calibration error is generated. When the CTC calibration counter count value is greater than or equal to 128 x CKLIM during decimation and a synchronization reference pulse signal is detected, CKERR is set, indicating that the current clock is too slow to calibrate to the desired frequency value. When ERRIE in CTC_CTL0 is set to 1, an interrupt is generated. The CKERR bit can be cleared to zero by writing a 1 to the ERRIC bit in CTC_INTC. 0: No clock calibration error occurred. 1: Clock calibration error occurred.
7:4	Reserved	RES	-	Reserved
3	EREFIF	R	0	Expect reference interrupt flag bit. When the CTC calibration clock counter counts to 0, a reference signal is detected and this bit is set by hardware. When EREFIE in CTC_CTL0 is set to 1, an interrupt is generated. The EREFIF bit can be cleared to zero by writing a 1 to the EREFIC bit in CTC_INTC. 0: No desired reference signal generation. 1: Expected reference signal generation.
2	ERRIF	R	0	Error interrupt flag bit. This bit is set by hardware when an error occurs. This bit is set whenever a TRIMERR, REFMISS, or CKERR error occurs. An interrupt is generated when ERRIE is set in CTC_CTL0. The ERRIF bit can be cleared to zero by writing a 1 to the ERRIC bit in CTC_INTC.

Bit	Name	R/W	Reset Value	Function
				0: No error occurred. 1: Error occurred.
1	CKWARNIF	R	0	Clock calibration warning interrupt flag bit. This bit is set by hardware when a clock calibration warning is generated. CKWARNIF is set when the CTC calibration counter count value is greater than or equal to 3xCKLIM and less than 128xCKLIM and a synchronization reference pulse signal is detected. This indicates that the current clock frequency is too slow or too fast, but can be calibrated to achieve the desired frequency value. When a clock calibration warning is generated, the TRIMVALUE value is incremented or decremented by 2. An interrupt is generated when CKWARNIE is set to 1 in CTC_CTL0. The CKWARNIF bit can be cleared to zero by writing a 1 to the CKWARNIC bit in CTC_INTC. 0: No clock calibration warning occurs. 1: Clock calibration warning occurs.
0	CKOKIF	R	0	Clock calibration success interrupt flag bit. This bit is set by hardware when clock calibration is successful. If a synchronization reference pulse signal is detected when the CTC calibration counter count value is less than 3 x CKLIM, CKOKIF is set. It indicates that the current clock frequency is normal and can be used without clock calibration by TRIMVALUE value. When CKOKIE in CTC_CTL0 is set to 1, an interrupt is generated. The CKOKIF bit can be cleared to zero by writing 1 to the CKOKIC bit in CTC_INTC. 0: Clock calibration unsuccessful. 1: Clock calibration successful.

28.3.4. Interrupt clear register (CTC_INTC)

Address offset:0x0C

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EREFIC	ERRIC	CKWARNIC	CKOKIC
												W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 4	Reserved	RES	-	Reserved

Bit	Name	R/W	Reset Value	Function
3	EREFIC	W	0	EREFIF interrupt clear bit. This bit can only be written by software, and a read operation returns 0. A write of 1 clears the EREFIF bit in CTC_STAT, and a write of 0 has no effect.
2	ERRIC	W	0	ERRIF interrupt clear bit. This bit can only be written by software, and a read operation returns 0. A write of 1 clears the ERRIF bit, TRIMERR bit, REFMISS bit, and CKERR bit from CTC_STAT, and a write of 0 has no effect.
1	CKWARNIC	W	0	CKWARNIF interrupt clear bit. This bit can only be written by software, and a read operation returns 0. A write of 1 clears the CKWARNIF bit in CTC_STAT; a write of 0 has no effect.
0	CKOKIC	W	0	CKOKIF interrupt clear bit. This bit can only be written by software, a read operation returns 0. A write of 1 clears the CKOKIF bit in CTC_STAT, a write of 0 has no effect.

28.3.5. CTC register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CTC_CTL0	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TRIMVALUE[5:0]							SWREFPUL	AUTOTRIM	CNTEN	Res	EREFIE	ERRIE	CKWARNIE	CKOKIE
	Reset value																		1	0	0	0	0	0	0	0	0	0		0	0	0	0
0x04	CTC_CTL1	REFPOL	Res	REFSEL[1:0]		Res	REFPSC[2:0]			CKLIM[7:0]								RLVALUE[15:0]															
	Reset value	0		1	0		0	0	0	0x22								0xBB7F														
0x08	CTC_SR	REFCAP[15:0]																REFDIR	Res	Res	Res	Res	TRIMERR	REFMISS	CKERR	Res	Res	Res	Res	EREFIF	ERRIF	CKWARNIF	CKOKIF
	Reset value	0																1					0	0	0					0	0	0	0
0x0C	CTC_INTC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EREFIC	ERRIC	CKWARNIC	CKOKIC
	Reset value																													0	0	0	0

29. Debug support (DBG)

29.1. Introduction

The MCUDBG module assists the debugger by providing the following features:

- Low Power Mode
- Provides clock control of Timer, Watchdog, I2C and CAN at breakpoints
- Control over trace pin assignment

The MCUDBG register also provides the chip ID code. This ID code can be accessed using the JTAG or SW debug interfaces, or by the user program.

29.1.1. Main features

- Supports sleep mode, stop mode and standby mode
- Control timer and watchdog to stop counting or continue counting when CPU enters HALT.
- Block I2C1 and I2C2 SMBUS timeout when CPU enters HALT.
- When the CPU enters HALT, it prevents the CAN receive register from being updated.
- Assign trace pins

29.2. Function description

29.2.1. Low-power mode is supported in debugging

Low power modes can be entered using WFI and WFE.

The MCU supports several low-power modes, which can turn off the CPU clock or reduce the power consumption of the CPU, respectively.

The kernel does not allow FCLK or HCLK to be turned off during debugging.These clocks are necessary for debugging operations, so they must work during debugging.The MCU uses a special way to allow the user to debug code in low-power mode.

To implement this feature, the debugger must first set some configuration registers to change the characteristics of the low-power mode.

- In sleep mode, the debugger must first reset the DBG_SLEEP bit of the DBGMCU_CR register. This will provide the same clock for HCLK as FCLK (the system clock configured by the code).
- In stop mode, the debugger must first set the DBG_STOP bit. This will activate the HSI8M clock, which provides the clock for FCLK and HCLK in stop mode.

29.3. Register baseaddr=0xE0042000

29.3.1. ID code (DBGMCU_IDCODE) (0xE004_2000)

Address offset：0x00

Reset value：0xXXXX XXXX

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

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

Bit	Name	R/W	Reset Value	Function
31: 0	REV_ID	R	x	MCU Product Emulator ID The value is stored in the FLASH information area.

29.3.2. Debug configuration register (DBGMCU_CR) (0xE004_2004)

This register is reset by a power-on reset of the VDDD domain; a system reset does not reset this register. The debugger can use this register when the kernel is in the reset state.

Address offset：0x04

Reset value：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	Res				DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM8_STOP	DBG_I2C2_SMBUS_TIMEOUT
	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**
DBG_I2C1_SMBUS_TIMEOUT	DBG_CAN_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE[1:0]		TRACE_IOEN	Res		DBG_STDBY	DBG_STOP	DBG_SLEEP
RW	RW	RW	RW	RW	RW	RW	RW	RW		RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Reserved	RES	-	Reserved
30	DBG_TIM11_STOP	RW	0	Control TIM11 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;
29	DBG_TIM10_STOP	RW	0	Control TIM10 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;
28	DBG_TIM9_STOP	RW	0	Control TIM9 counting halt when the CPU core is in halt state.

Bit	Name	R/W	Reset Value	Function
				0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;
27	DBG_TIM14_STOP	RW	0	Control TIM14 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;
26	DBG_TIM13_STOP	RW	0	Control TIM13 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;
25	DBG_TIM12_STOP	RW	0	Control TIM12 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;
24:21	Reserved	RES	-	Reserved
20	DBG_TIM7_STOP	RW	0	Control TIM7 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;
19	DBG_TIM6_STOP	RW	0	Control TIM6 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;
18	DBG_TIM5_STOP	RW	0	Control TIM5 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;

Bit	Name	R/W	Reset Value	Function
17	DBG_TIM8_STOP	RW	0	Control TIM8 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counting is normal; 1: When CPU halt, counter clock disable, stop counting, output disable;
16	DBG_I2C2_SMBUS_TIMEOUT	RW	0	Control the I2C2 SMBUS timeout mode halt when the CPU core is in halt state. 0: When CPU halt, I2C2 is the same as normal mode; 1: When CPU halt, I2C2 SMBUS timeout function is disabled;
15	DBG_I2C1_SMBUS_TIMEOUT	RW	0	Control the I2C1 SMBUS timeout mode halt when the CPU core is in halt state. 0: When CPU halt, I2C2 is the same as normal mode; 1: When CPU halt, I2C2 SMBUS timeout function is disabled;
14	DBG_CAN_STOP	RW	0	Control CAN halt when the CPU core is in halt state. 0: When CPU halt, CAN is the same as normal mode; 1: When CPU halt, CAN receive register is disabled;
13	DBG_TIM4_STOP	RW	0	Control TIM4 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counts normally; 1: When CPU halt, counter clock disable, stop counting;
12	DBG_TIM3_STOP	RW	0	Control TIM3 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counts normally; 1: When CPU halt, counter clock disable, stop counting;
11	DBG_TIM2_STOP	RW	0	Control TIM2 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counts normally; 1: When CPU halt, counter clock disable, stop counting;
10	DBG_TIM1_STOP	RW	0	Control TIM1 counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counts normally; 1: When CPU halt, counter clock disable, stop counting;

Bit	Name	R/W	Reset Value	Function
9	DBG_WWDG_STOP	RW	0	Control WWDG counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counts normally; 1: When CPU halt, counter clock disable, stop counting;
8	DBG_IWDG_STOP	RW	0	Control IWDG counting halt when the CPU core is in halt state. 0: When the CPU halt, the counter clock is enabled and counts normally; 1: When CPU halt, counter clock disable, stop counting;
7:6	TRACE_MODE[1:0]	RW	0	Trace pin assignment. 00: trace pins use asynchronous mode; 01: trace pins use synchronous mode and TRACEDATA length is 1; 10: trace pins use synchronous mode and the TRACEDATA length is 2; 11: trace pins use synchronous mode and have a TRACEDATA length of 4;
5	TRACE_IOEN	RW	0	Trace pin assignment enable. 0: no trace pin assignment (default state); 1: Trace pin assignment as defined by TRACE_MODE;
4:3	Reserved	RES	-	Reserved
2	DBG_STDBY	RW	0	Debugging standby mode. 0: (FCLK off, HCLK off), normal standby mode processing; 1: (FCLK on, HCLK on), and the system clock is provided by HSI8M. the MCU generates a system reset to exit the standby mode as it does after power-on reset.
1	DBG_STOP	RW	0	Debug stop mode. 0: (FCLK off, HCLK off). In stop mode, both HCLK and FCLK are turned off. When exiting from stop mode, the clock configuration is the same as after reset (system clock is HSI8M). Subsequently, software needs to reconfigure the clock controller to enable the PLL, HSE, etc. 1: (FCLK on, HCLK on). When entering stop mode, the internal system clock source is not turned off and FCLK and HCLK are present. When exiting stop mode, the software needs to be reconfigured if the clock control needs to be changed.
0	DBG_SLEEP	RW	0	Debugging sleep mode.

Bit	Name	R/W	Reset Value	Function
				0: (FCLK on, HCLK off). In sleep mode, FCLK is supplied by the originally configured system clock and HCLK is off. Since sleep mode does not reset the configured clock system, the software does not need to reconfigure the clock after exiting from sleep mode. 1: (FCLK on, HCLK on). In sleep mode, both the FCLK and HCLK clocks are provided by the originally configured system clock.

29.3.3. DBG register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004_2000	DBG_IDCODE	REV_ID[31:0]																															
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x04	DBG__CR	Reserved	DBG_TIM11_STO	DBG_TIM10_STO	DBG_TIM9_STO	DBG_TIM14_STO	DBG_TIM13_STO	DBG_TIM12_STO	Reserved				DBG_TIM7_STO	DBG_TIM6_STO	DBG_TIM5_STO	DBG_TIM8_STO	DBG_I2C2_SMB	DBG_I2C1_SMB	DBG_CAN_STO	DBG_TIM4_STO	DBG_TIM3_STO	DBG_TIM2_STO	DBG_TIM1_STO	DBG_WWDG_STO	DBG_IWDG_STO	TRACE_MODE[1:0]		TRACE_IOEN	Reserved		DBG_STDBY	DBG_STOP	DBG_SLLEP
	Reset Value		0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0

30. Version History

Version	Date	Updated Record
V1.0	2023.08.30	First version
V1.1	2023.10.26	Updated Table 1-1
V1.2	2024.02.29	Updated FLASH access control register(FLASH_ACR)
V1.3	2024.10.30	Updated 11. Interrupts and events→11.1.1. Main features

Puya Semiconductor Co., Ltd.