MG32F157xx

ARM® 32-bit Cortex™-M3 MCU

Reference Manual

Preliminary

2022.12

Contents

List of Figures

List of Tables

Introduction

This reference manual provides detailed information on how to use memory and peripherals of MG32F157xx series microcontroller , including internal structure of each function module, description of all possible functions, usage of various working modes and register configuration to help users solve application problems

MG32F157xx series microcontrollers have different memory capacities, packages and peripheral configurations. The configurations mentioned in this reference manual are only the highest of the series.

For the memory capacity, package and peripheral configuration, electrical and physical performance parameters of the MG32F157xx series, please refer to the data manual of the specific model of the series.

Download the relevant information:

http://www.megawin.com.tw

This document is only used to help users understand and use the product, and does not assume any loss or damage caused by the use of any information in this document or any incorrect use of the product!

1 Document convention

1.1 The list of abbreviations used in the register description table

The following abbreviations are used in the description of the register:

Tab 1.1-1 Abbreviated list

Abbreviation	Full name	Description
R	read-only	Software can only read this bit
W	write-only	Software can only write this bit
RW	read/write	Software can read and write this bit
RC_W0	read/clear write0	Software can read this bit or clear it by writing '0'. Writing '1' has no effect on this bit
RC_W1	read/clear write1	Software can read this bit or clear it by writing '1'. Writing '0' has no effect on this bit
RC_R	read/clear by read	Software can read this bit. When this bit is read, it will be cleared automatically. Writing has no effect on this bit
RS_R	read/set by read	Software can read this bit. When this bit is read, it is automatically set. Writing has no effect on this bit.
RS	read/set	Software can read or set this bit, write '0' has no effect on this bit
S	set	The software can only set this bit. Writing '0' has no effect on this bit
T	toggle	The software can only flip this bit by writing '1'. Writing '0' has no effect on this bit
Res.	Reserved	Reserved bit.

2 Memory and bus architecture

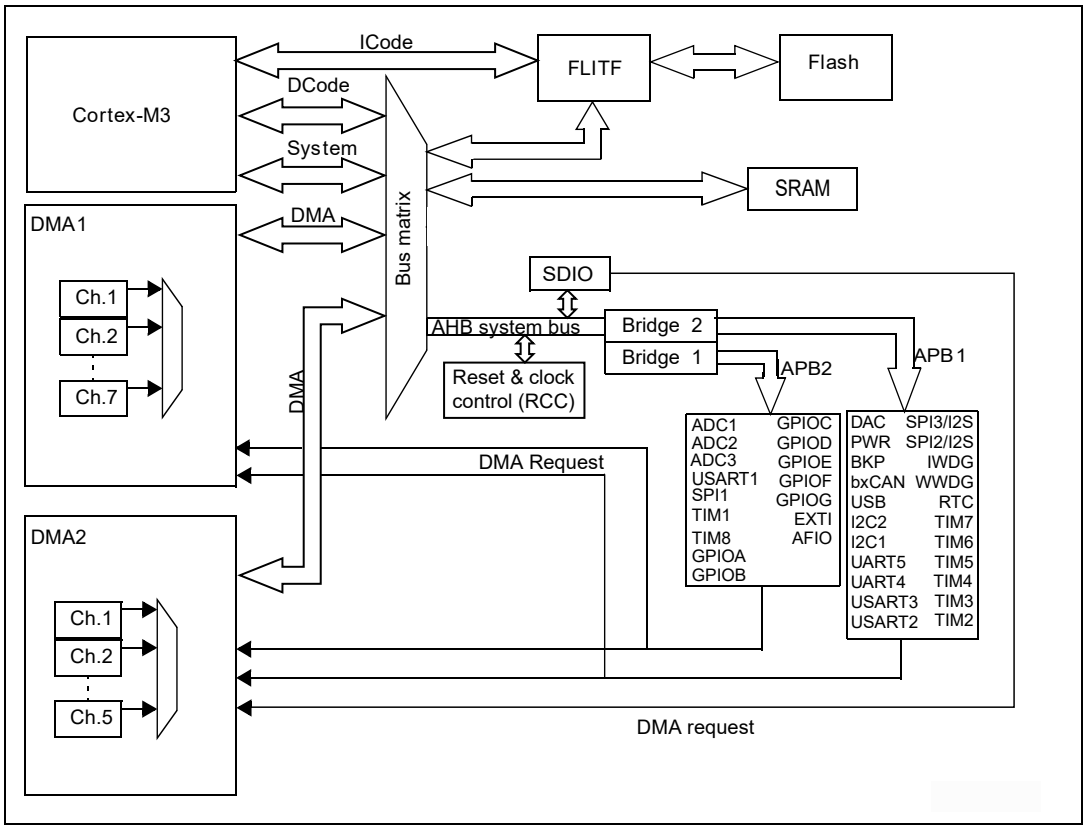
2.1 System architecture

The main system consists of:

- Four masters:
  - Cortex™-M3 core DCode bus (D-bus) and System bus (S-bus)
  - GP-DMA 1/2 (general-purpose DMA 1/2)
- three slaves:
  - Internal SRAM
  - Internal Flash memory
  - AHB to APBx (APB1 or APB2), which connect all the APB peripherals

These are interconnected using a multilayer AHB bus architecture

Fig 2.1-1 System architecture



1. ICode bus
   This bus connects the Instruction bus of the Cortex™-M3 core to the Flash memory instruction interface. Prefetching is performed on this bus.

2. DCode bus
   This bus connects the DCode bus (literal load and debug access) of the Cortex™-M3 core to the Flash memory Data interface.

3. System bus
   This bus connects the system bus of the Cortex™-M3 core (peripherals bus) to a BusMatrix which manages the arbitration between the core and the DMA.

4. DMA bus
   This bus connects the AHB master interface of the DMA to the BusMatrix which manages the access of CPU DCode and DMA to SRAM, Flash memory and peripherals.

5. BusMatrix
   the BusMatrix is composed of four masters (CPU DCode, System bus, DMA1 bus and DMA2 bus) and three slaves (FLITF, SRAM and AHB2APB bridges).
   AHB peripherals are connected on system bus through a BusMatrix to allow DMA access.
6. AHB/APB bridges (APB)
   The two AHB/APB bridges provide full synchronous connections between the AHB and the 2 APB buses. APB1 is limited to 48 MHz, APB2 operates at full speed (up to 96 MHz depending on the device). Refer to Tab:2.3-1 for the address mapping of the peripherals connected to each bridge. After each device reset, all peripheral clocks are disabled (except for the SRAM and FLITF). Before using a peripheral you have to enable its clock in the RCC_AHBENR, RCC_APB2ENR or RCC_APB1ENR register.

Note: *When a 16- or 8-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 16- or 8-bit data to feed the 32-bit vector.*

2.2 Memory organization

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space. The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word’s least significant byte and the highest numbered byte the most significant.
The addressable memory space is divided into 8 main blocks, each of 512 MB.
All the memory areas that are not allocated to on-chip memories and peripherals are considered “Reserved”.

2.3 Memory map

Tab: 2.3-1 gives the boundary addresses of the peripherals available in MG32F157xx .

Tab 2.3-1 Register boundary addresses

Boundary address	Peripheral	Register map
AHB Bus		
0x4002 6000 - 0x4002 63FF	AES	
0x4002 5000 - 0x4002 53FF	TRNG	
0x4002 3000 - 0x4002 33FF	CRC	
0x4002 2000 - 0x4002 23FF	Flash memory interface	
0x4002 1400 - 0x4002 1FFF	Reserved	
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC	
0x4002 0800 - 0x4002 0FFF	Reserved	
0x4002 0400 - 0x4002 07FF	DMA2	
0x4002 0000 - 0x4002 03FF	DMA1	
0x4001 8400 - 0x4001 7FFF	Reserved	
0x4001 8000 - 0x4001 83FF	SDIO	
APB2 Bus		
0x4001 4000 - 0x4001 7FFF	QSPI	
0x4001 3C00 - 0x4001 3FFF	ADC3	
0x4001 3800 - 0x4001 3BFF	USART1	
0x4001 3400 - 0x4001 37FF	TIM8 timer	
0x4001 3000 - 0x4001 33FF	SPI1	
0x4001 2C00 - 0x4001 2FFF	TIM1 timer	
0x4001 2800 - 0x4001 2BFF	ADC2	
0x4001 2400 - 0x4001 27FF	ADC1	
0x4001 2000 - 0x4001 23FF	GPIO Port G	
0x4001 2000 - 0x4001 23FF	GPIO Port F	
0x4001 1800 - 0x4001 1BFF	GPIO Port E	
0x4001 1400 - 0x4001 17FF	GPIO Port D	
0x4001 1000 - 0x4001 13FF	GPIO Port C	
0X4001 0C00 - 0x4001 0FFF	GPIO Port B	
0x4001 0800 - 0x4001 0BFF	GPIO Port A	
0x4001 0400 - 0x4001 07FF	EXTI	
0x4001 0000 - 0x4001 03FF	AFIO	
APB1 Bus		
0x4000 7800 - 0x4000FFFF	Reserved	
0x4000 7400 - 0x4000 77FF	DAC	
0x4000 7000 - 0x4000 73FF	Power control PWR	
0x4000 6C00 - 0x4000 6FFF	Backup registers (BKP)	
0x4000 6800 - 0x4000 6BFF	Reserved	
0x4000 6400 - 0x4000 67FF	bxCAN1	
0x4000 6000 - 0x4000 63FF	Shared USB/CAN SRAM 512 bytes	
0x4000 5C00 - 0x4000 5FFF	USB device FS registers	
0x4000 5800 - 0x4000 5BFF	I2C2	
0x4000 5400 - 0x4000 57FF	I2C1	
0x4000 5000 - 0x4000 53FF	UART5	
0x4000 4C00 - 0x4000 4FFF	UART4	
0x4000 4800 - 0x4000 4BFF	USART3	
0x4000 4400 - 0x4000 47FF	USART2	

(Continuation)

Boundary address	Peripheral	Register map
0x4000 4000 - 0x4000 3FFF	Reserved	
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S3	
0x4000 3800 - 0x4000 3BFF	SPI2/I2S3	
0x4000 3400 - 0x4000 37FF	Reserved	
0x4000 3000 - 0x4000 33FF	Independent watchdog(IWDG)	
0x4000 2C00 - 0x4000 2FFF	Window watchdog(WWDG)	
0x4000 2800 - 0x4000 2BFF	RTC	
0x4000 1800 - 0x4000 27FF	Reserved	
0x4000 1400 - 0x4000 17FF	TIM7 timer	
0x4000 1000 - 0x4000 13FF	TIM6 timer	
0x4000 0C00 - 0x4000 0FFF	TIM5 timer	
0x4000 0800 - 0x4000 0BFF	TIM4 timer	
0x4000 0400 - 0x4000 07FF	TIM3 timer	
0x4000 0000 - 0x4000 03FF	TIM2 timer	

2.3.1 Embedded SRAM

MG32F157xx features up to 96 Kbytes of static SRAM. It can be accessed as bytes, half-words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000。

2.4 Bit banding

The Cortex™-M3 memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.
MG32F157xx both peripheral registers and SRAM are mapped in a bit-band region. This allows single bit-band write and read operations to be performed. The operations are only available for Cortex ® -M3 accesses, not from other bus masters (e.g. DMA).

$$bit\_word\_addr = bit\_band\_base + (byte\_offset \times 32) + (bit\_number \times 4)$$

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

Example:
The following example shows how to map bit 2 of the byte located at SRAM address 0x20000300 in the alias region:
0x22006008 = 0x22000000 + (0x300 × 32) + (2 × 4).
Writing to address 0x22006008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM address 0x20000300

Reading address 0x22006008 returns the value (0x01 or 0x00) of bit 2 of the byte at SRAM address 0x20000300 (0x01: bit set; 0x00: bit reset)

2.5 Embedded Flash memory

The high-performance Flash memory module has the following key features:

- density of up to 512 Kbytes, the Flash memory is organized as a main block and an information block:
  - main block: up to 64 Kb × 64 bits divided into 256 pages of 2 Kbytes each for devices, see Table2.5-1
  - information block: 258 × 64 bits ,see Table2.5-1
- The Flash memory interface (FLITF) features:
  - Read interface with prefetch buffer (2x64-bit words)
  - Option byte Loader
  - Flash Program / Erase operation
  - Read / Write protection

Tab 2.5-1 Flash module organization

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbytes
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbytes
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbytes
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbytes
	...	...	...
	Page 255	0x0809 5400 - 0x0809 57FF	1 Kbytes
Information block	System memory	0x1FFF E800 - 0x1FFF EBFF	1 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Reading the Flash memory

Flash memory instructions and data access are performed through the AHB bus. The prefetch block is used for instruction fetches through the ICode bus. Arbitration is performed in the Flash memory interface, and priority is given to data access on the DCode bus.
Read accesses can be performed with the following configuration options:

- Latency: number of wait states for a read operation programmed on-the-fly
- Prefetch buffer (2 x 64-bit blocks): it is enabled after reset; a whole block can be replaced with a

single read from the Flash memory as the size of the block matches the bandwidth of the Flash memory. Thanks to the prefetch buffer, faster CPU execution is possible as the CPU fetches one word at a time with the next word readily available in the prefetch buffer

- Half cycle: for power optimization

Note:

1. *These options have to be used in accordance with the Flash memory access time. The wait states represent the ratio of the SYSCLK (system clock) period to the Flash memory access time:*
   0 wait states, if 0 < SYSCLK ⩽ 48MHz
   1 wait states, if 48MHz < SYSCLK ⩽ 96MHz
2. *Half cycle configuration is not available in combination with a prescaler on the AHB. The system clock (SYSCLK) should be equal to the HCLK clock. This feature can therefore be used only with a low-frequency clock of 8 MHz or less. It can be generated from the HSI or the HSE but not from the PLL.*
3. *The prefetch buffer must be kept on when using a prescaler different from 1 on the AHB clock.*
4. *The prefetch buffer must be switched on/off only when SYSCLK is lower than 24 MHz and no prescaler is applied on the AHB clock (SYSCLK must be equal to HCLK). The prefetch buffer is usually switched on/off during the initialization routine, while the microcontroller is running on the internal 8 MHz RC (HSI) oscillator.*
5. *Using DMA: DMA accesses Flash memory on the DCode bus and has priority over ICode instructions. The DMA provides one free cycle after each transfer. Some instructions can be performed together with DMA transfer.*

Programming and erasing the Flash memory

The Flash memory can be programmed 16 bits (half words) at a time.
For write and erase operations on the Flash memory (write/erase), the internal RC oscillator (HSI) must be ON. The Flash memory erase operation can be performed at page level or on the whole Flash area (mass-erase). The mass-erase does not affect the information blocks.
To ensure that there is no over-programming, the Flash Programming and Erase Controller blocks are clocked by a fixed clock.
The End of write operation (programming or erasing) can trigger an interrupt. This interrupt can be used to exit from WFI mode, only if the FLITF clock is enabled. Otherwise, the interrupt is served only after an exit from WFI.

Flash access control register (FLASH_ACR)

Register	Address offset	Access	Reset value	Description
FLASH_ACR	0x00	RW	0x0000_0030	Flash access control register

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

Flash access control register (FLASH_ACR)bit description

Bit	Access	Description
[31:6]	R	Reserved,must be kept at reset value
[5]	R	Reserved
[4:3]	RW	Reserved
[2:0]	RW	**LATENCY:** Latency These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time. 0 wait state, if 0 < SYSCLK ⩽ 48 MHz 1 wait state, if 48 MHz < SYSCLK ⩽ 96 MHz

2.6 Boot configuration

In the MG32F157xx , 3 different boot modes can be selected through BOOT[1:0] pins

Tab 2.6-1 Boot modes

Boot mode selection pins		Boot modes	Aliasing
BOOT1	BOOT0		
X	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

The values on the BOOT pins are latched on the 4th rising edge of SYSCLK after a reset. It is up to the user to set the BOOT1 and BOOT0 pins after Reset to select the required boot mode.

The BOOT pins are also re-sampled when exiting from Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory starting from 0x0000 0004.

Due to its fixed memory map, the code area starts from address 0x0000_0000 (accessed through the ICode/DCode buses) while the data area (SRAM) starts from address 0x2000_0000 (accessed through the system bus). The Cortex-M3 CPU always fetches the reset vector on the ICode bus, which implies to have the boot space available only in the code area (typically, Flash memory). MG32F157xx microcontrollers implement a special mechanism to be able to boot also from SRAM and not only from main Flash memory and System memory.

Depending on the selected boot mode, main Flash memory, system memory or SRAM is accessible as follows:

- Boot from main Flash memory:
  The main Flash memory is aliased in the boot memory space (0x0000_0000), but still accessible from its original memory space (0x0800_0000). In other words, the Flash memory contents can be accessed starting from address 0x0000_0000 or 0x0800_0000.
- Boot from system memory:

the system memory is aliased in the boot memory space (0x0000_0000), but still accessible from its original memory space (0x1FFF_E800).

- Boot from the embedded SRAM:
  SRAM is accessible only at address 0x2000_0000.

2.7 Embedded boot loader

The embedded boot loader is located in the System memory, programmed during production. It is used to reprogram the Flash memory with one of the available serial interfaces:The bootoader is activated through the USART1 interface.

3 FLASH

3.1 FLASH Features

- up to 256 Kbytes of Flash memory
- Memory organization:
  - Main memory block:
    32 Kbits × 64 bits
  - Information block:
    258 × 64 bits

  Flash memory interface (FLITF) features:
- Read interface with prefetch buffer (2 × 64-bit words)
- Option byte Loader
- Flash Program / Erase operation
- Read / Write protection
- Low-power mode

3.2 Flash module organization

The memory organization is based on a main memory block containing 256 pages of 1 Kbyte , and an information block as shown in Table 3.2-1 .

Tab 3.2-1 Flash module organization

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbytes
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbytes
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbytes
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbytes
	...	...	...
	Page 255	0x0809 5400 - 0x0809 57FF	1 Kbytes
Information block	System memory	0x1FFF E800 - 0x1FFF EBFF	1 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

The Flash memory is organized as 32-bit wide memory cells that can be used for storing both code and data constants. The Flash module is located at a specific base address in the memory map of each MG32F157xx microcontroller type.

The information block is divided into two parts:

- System memory is used to boot the device in System memory boot mode. The area contains the boot loader which is used to reprogram the Flash memory using the USART1 serial interface.
- Option bytes

Write operations to the main memory block and the option bytes are managed by an embedded Flash Program/Erase Controller (FPEC). The high voltage needed for Program/Erase operations is internally generated.

The main Flash memory can be protected against different types of unwanted access (read/write/erase). There are two types of protection:

- Page Write Protection
- Read Protection

During a write operation to the Flash memory, any attempt to read the Flash memory will stall the bus. The read operation will proceed correctly once the write operation has completed. This means that code or data fetches cannot be made while a write/erase operation is ongoing.

For write and erase operations on the Flash memory (write/erase), the internal RC oscillator (HSI) must be ON.

The Flash memory can be programmed and erased using in-circuit programming and in- application programming.

Note: *In the low-power modes, all Flash memory accesses are aborted.*

3.3 Reading/programming the embedded Flash memory

3.3.1 Read operation

The embedded Flash module can be addressed directly, as a common memory space. Any data read operation accesses the content of the Flash module through dedicated read senses and provides the requested data.

The read interface consists of a read controller on one side to access the Flash memory and an AHB interface on the other side to interface with the CPU. The main task of the read interface is to generate the control signals to read from the Flash memory and to prefetch the blocks required by the CPU. The prefetch block is only used for instruction fetches over the I-Code bus. The Literal pool is accessed over the D-Code bus. Since these two buses have the same Flash memory as target, D-code bus accesses have priority over prefetch accesses.

3.3.2 Instruction fetch

The Cortex-M3 fetches the instruction over the I-Code bus and the literal pool (constant/data) over the D-code bus. The prefetch block aims at increasing the efficiency of I-Code bus accesses.

Prefetch buffer

The prefetch buffer is 2 blocks wide where each block consists of 8 bytes. The prefetch blocks are direct-mapped. A block can be completely replaced on a single read to the Flash memory as the size of the block matches the bandwidth of the Flash memory.

The implementation of this prefetch buffer makes a faster CPU execution possible as the CPU fetches one word at a time with the next word readily available in the prefetch buffer. This implies that the acceleration ratio will be of the order of 2 assuming that the code is aligned at a 64-bit boundary for the jumps.

Prefetch controller

The prefetch controller decides to access the Flash memory depending on the available space in the prefetch buffer. The Controller initiates a read request when there is at least one block free in the prefetch buffer.

After reset, the state of the prefetch buffer is on.
The prefetch buffer should be switched on/off only when SYSCLK is lower than 24 MHz and no prescaler is applied on the AHB clock (SYSCLK must be equal to HCLK). The prefetch buffer is usually switched on/off during the initialization routine, while the microcontroller is running on the internal 8 MHz RC (HSI) oscillator.

Note: *The prefetch buffer must be kept on (FLASH_ACR[4]=’1’) when using a prescaler different from 1 on the AHB clock.*

In case of non-availability of a high frequency clock in the system, Flash memory accesses can be

made on a half cycle of HCLK (AHB clock), the frequency of HCLK permitting (half- cycle access can only be used with a low-frequency clock of less than 8 MHz that can be obtained with the use of HSI or HSE but not of PLL). This mode can be chosen by setting a control bit in the Flash access control register.

Note: *Half-cycle access cannot be used when there is a prescaler different from 1 on the AHB clock.*

Access time tuner

In order to maintain the control signals to read the Flash memory, the ratio of the prefetch controller clock period to the access time of the Flash memory has to be programmed in the Flash access control register. This value gives the number of cycles needed to maintain the control signals of the Flash memory and correctly read the required data. After reset, the value is zero and only one cycle is required to access the Flash memory.

3.3.3 D-Code interface

The D-Code interface consists of a simple AHB interface on the CPU side and a request generator to the Arbiter of the Flash access controller. D-code accesses have priority over prefetch accesses. This interface uses the Access Time Tuner block of the prefetch buffer.

3.3.4 Flash access controller

Mainly, this block is a simple arbiter between the read requests of the prefetch/I-code and D- Code interfaces.

D-Code interface requests have priority over I-Code requests.

3.4 Flash program and erase controller (FPEC)

The FPEC block handles the program and erase operations of the Flash memory. The FPEC consists of seven 32-bit registers.

- FPEC key register (FLASH_KEYR)
- Option byte key register (FLASH_OPTKEYR)
- Flash control register (FLASH_CR)
- Flash status register (FLASH_SR)
- Flash address register (FLASH_AR)
- Option byte register (FLASH_OBR)
- Write protection register (FLASH_WRPR)

An ongoing Flash memory operation will not block the CPU as long as the CPU does not access the Flash memory.

3.4.1 Key values

The key values are as follows:

- RDPRT key = 0xA5

- KEY1 = 0x45670123
- KEY2 = 0xCDEF89AB

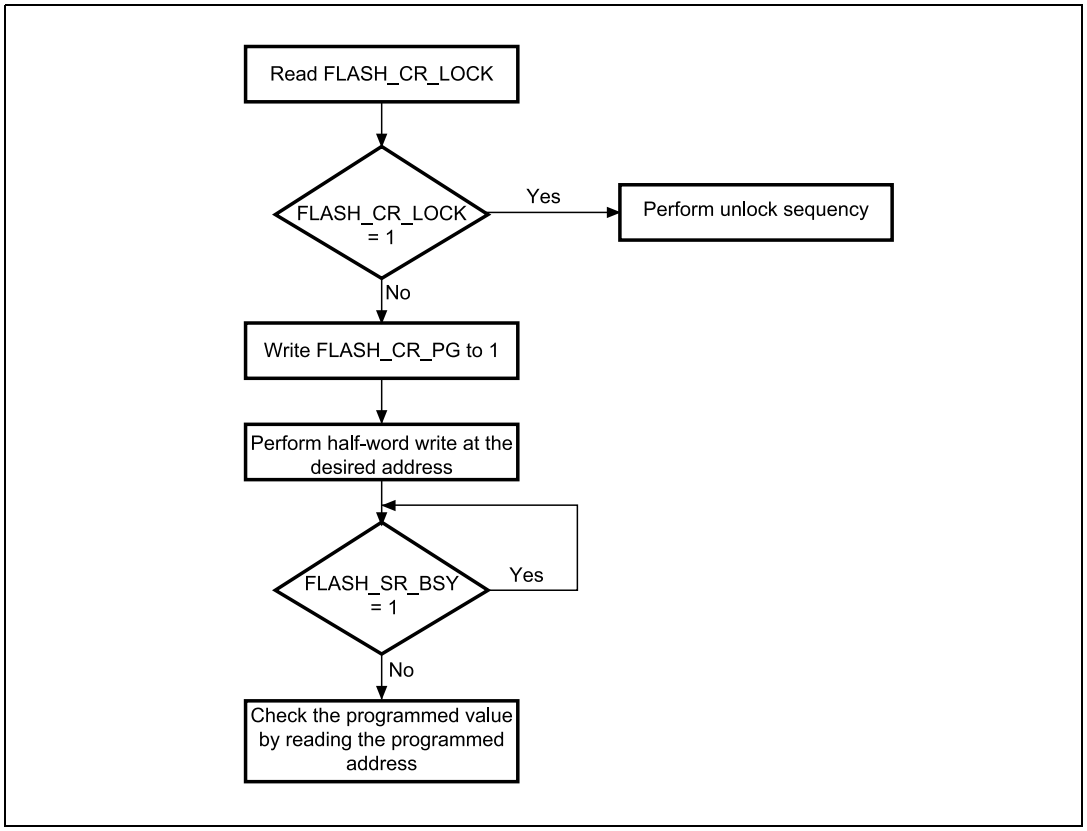
3.4.2 Unlocking the Flash memory

After reset, the FPEC block is protected. The FLASH_CR register is not accessible in write mode. An unlocking sequence should be written to the FLASH_KEYR register to open up the FPEC block. This sequence consists of two write cycles, where two key values (KEY1 and KEY2) are written to the FLASH_KEYR address . Any wrong sequence locks up the FPEC block and FLASH_CR register until the next reset. Also a bus error is returned on a wrong key sequence. This is done after the first write cycle if KEY1 does not match, or during the second write cycle if KEY1 has been correctly written but KEY2 does not match. The FPEC block and FLASH_CR register can be locked by the user's software by writing the LOCK bit of the FLASH_CR register to 1. In this case, the FPEC can be unlocked by writing the correct sequence of keys into FLASH_KEYR.

3.4.3 Main Flash memory programming

The main Flash memory can be programmed 16 bits at a time. The program operation is started when the CPU writes a half-word into a main Flash memory address with the PG bit of the FLASH_CR register set. Any attempt to write data that are not half-word long will result in a bus error response from the FPEC. If a read/write operation is initiated during programming, (BSY bit set), the CPU stalls until the ongoing main Flash memory programming is over.

Fig 3.4-1 Programming procedure



Standard programming

In this mode the CPU programs the main Flash memory by performing standard half-word write operations. The PG bit in the FLASH_CR register must be set. FPEC preliminarily reads the value at the addressed main Flash memory location and checks that it has been erased. If not, the program operation is skipped and a warning is issued by the PGERR bit in FLASH_SR register (the only exception to this is when 0x0000 is programmed. In this case, the location is correctly programmed to 0x0000 and the PGERR bit is not set). If the addressed main Flash memory location is write-protected by the FLASH_WRPR register, the program operation is skipped and a warning is issued by the WRPRTERR bit in the FLASH_SR register. The end of the program operation is indicated by the EOP bit in the FLASH_SR register.

The main Flash memory programming sequence in standard mode is as follows:

- Check that no main Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
- Set the PG bit in the FLASH_CR register.
- Perform the data write (half-word) at the desired address.
- Wait for the BSY bit to be reset.
- Read the programmed value and verify

Note: *The registers are not accessible in write mode when the BSY bit of the FLASH_SR register is set.*

3.4.4 Flash memory erase

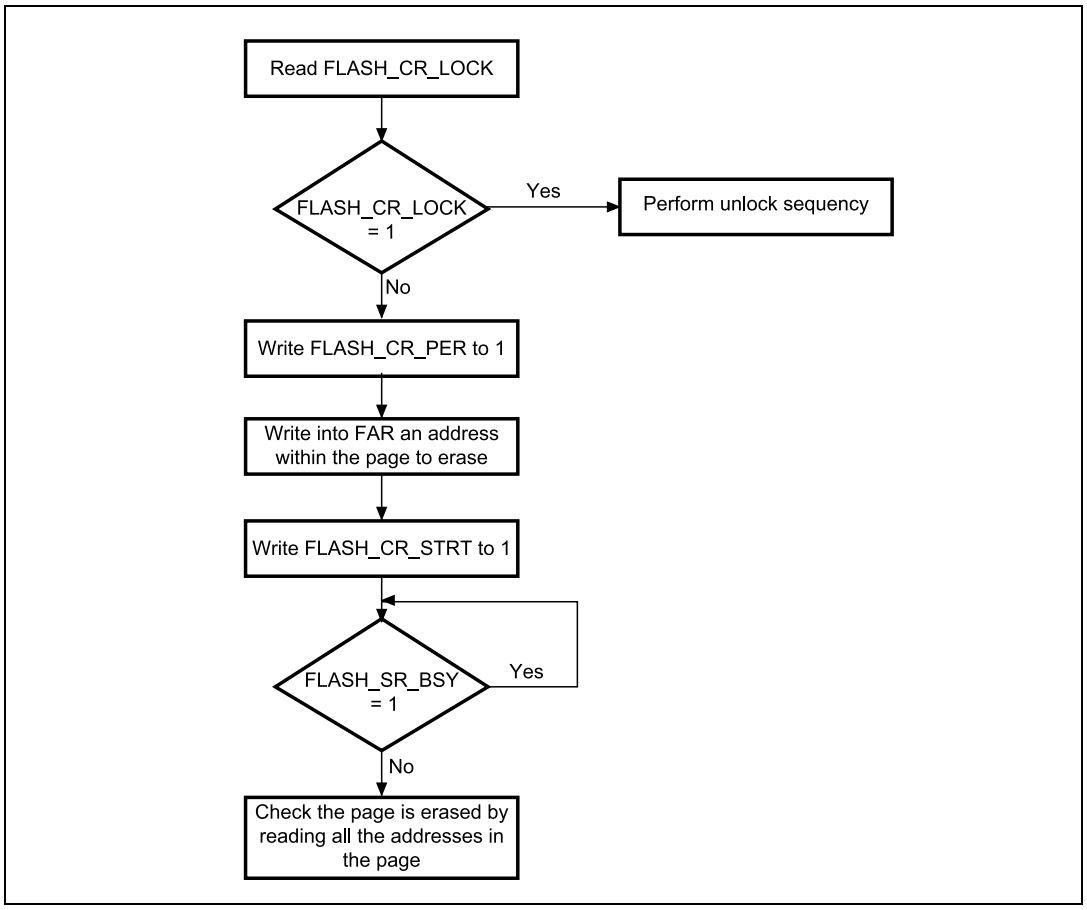
The Flash memory can be erased page by page or completely (Mass Erase).

Page Erase

A page of the Flash memory can be erased using the Page Erase feature of the FPEC. To erase a page, the procedure below should be followed:

- Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_CR register
- Set the PER bit in the FLASH_CR register
- Program the FLASH_AR register to select a page to erase
- Set the STRT bit in the FLASH_CR register
- Wait for the BSY bit to be reset
- Read the erased page and verify

Fig 3.4-2 Flash memory Page Erase procedure

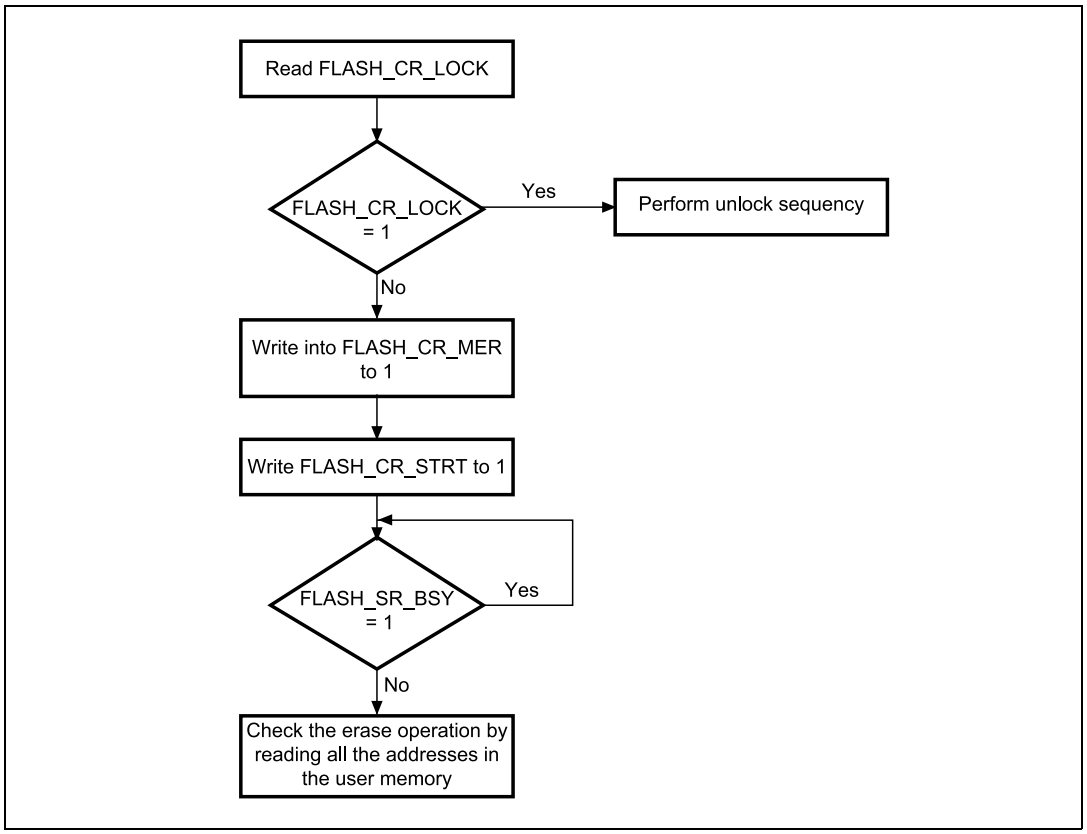


Mass Erase

The Mass Erase command can be used to completely erase the user pages of the Flash memory. The information block is unaffected by this procedure. The following sequence is recommended:

- Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register
- Set the MER bit in the FLASH_CR register
- Set the STRT bit in the FLASH_CR register
- Wait for the BSY bit to be reset
- Read all the pages and verify

Fig 3.4-3 Flash memory Mass Erase procedure



3.4.5 Option byte programming

The option bytes are programmed differently from normal user addresses. The number of option bytes is limited to 8 (4 for write protection, 1 for read protection, 1 for configuration and 2 for user data storage). After unlocking the FPEC, the user has to authorize the programming of the option bytes by writing the same set of KEYS (KEY1 and KEY2) to the FLASH_OPTKEYR register to set the OPTWRE bit in the FLASH_CR register (refer to Section 3.4.1 for key values). Then the user has to set the OPTPG bit in the FLASH_CR register and perform a half-word write operation at the desired Flash address.

FPEC preliminarily reads the value of the addressed option byte and checks that it has been erased. If not, the program operation is skipped and a warning is issued by the PGERR bit in the FLASH_SR register. The end of the program operation is indicated by the EOP bit in the FLASH_SR register.

The FPEC takes the LSB and automatically computes the MSB (which is the complement of the LSB) and starts the programming operation. This guarantees that the option byte and its complement are always correct.

The sequence is as follows:

- Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register.
- Unlock the OPTWRE bit in the FLASH_CR register.

- Set the OPTPG bit in the FLASH_CR register
- Write the data (half-word) to the desired address
- Wait for the BSY bit to be reset.
- Read the programmed value and verify.

When the Flash memory read protection option is changed from protected to unprotected, a Mass Erase of the main Flash memory is performed before reprogramming the read protection option. If the user wants to change an option other than the read protection option, then the mass erase is not performed. The erased state of the read protection option byte protects the Flash memory.

Erase procedure

The option byte erase sequence (OPTERASE) is as follows:

- Check that no Flash memory operation is ongoing by reading the BSY bit in the FLASH_SR register
- Unlock the OPTWRE bit in the FLASH_CR register
- Set the OPTER bit in the FLASH_CR register
- Set the STRT bit in the FLASH_CR register
- Wait for BSY to reset
- Read the erased option bytes and verify

3.5 Protections

The user area of the Flash memory can be protected against read by untrusted code. The pages of the Flash memory can also be protected against unwanted write due to loss of program counter contexts. The write-protection granularity is then of:

- two pages

3.5.1 Read protection

The read protection is activated by setting the RDP option byte and then, by applying a system reset to reload the new RDP option byte.

Note: *If the read protection is set while the debugger is still connected through SWD, apply a POR (power-on reset) instead of a system reset (without debugger connection).*

Once the protection byte has been programmed:

- Main Flash memory read access is not allowed except for the user code (when booting from main Flash memory itself with the debug mode not active).
- Pages 0-1 are automatically write-protected. The rest of the memory can be programmed by the code executed from the main Flash memory (for IAP, constant storage, etc.), but it is protected against write/erase (but not against mass erase) in debug mode or when booting from the embedded SRAM.
- All features linked to loading code into and executing code from the embedded SRAM are still active (SWD and boot from embedded SRAM) and this can be used to disable the read protection. When the read protection option byte is altered to a memory-unprotect value, a mass erase is

performed.

- When booting from the embedded SRAM, Flash memory accesses through the code and through data read using DMA1 and DMA2 are not allowed.
- Flash memory access through data read using SWV (serial wire viewer), SWD (serial wire debug), ETM and boundary scan are not allowed.

The Flash memory is protected when the RDP option byte and its complement contain the pair of values shown in Tab:3.5-1

Tab 3.5-1 Flash memory protection status

RDP byte value	RDP complement value	Read protection status
0xFF	0xFF	Protected
RDPRT	Complement of RDP byte	Not protected
Any value	Not the complement value of RDP	Protected

Note: *Erasing the option byte block will not trigger a mass erase as the erased value (0xFF) corresponds to a protected value.*

Unprotection

To disable the read protection from the embedded SRAM:

- Erase the entire option byte area. As a result, the read protection code (RDP) will be 0xFF. At this stage the read protection is still enabled.
- Program the correct RDP code 0x00A5 to unprotect the memory. This operation first forces a Mass Erase of the main Flash memory.
- Reset the device (POR Reset) to reload the option bytes (and the new RDP code) and, to disable the read protection.

Note: *The read protection can be disabled using the boot loader (in this case only a System Reset is necessary to reload the option bytes).*

3.5.2 Write protection

In high-density and connectivity line devices, from page 0 to page 61, write protection is implemented with a granularity of two pages at a time. The remaining memory block (from page 62 to page 255) is write-protected at once.

If a program or an erase operation is performed on a protected page, the Flash memory returns a protection error flag on the Flash memory Status Register (FLASH_SR).

The write protection is activated by configuring the WRP[3:0] option bytes, and then by applying a system reset to reload the new WRPx option bytes

Unprotection

To disable the write protection, two application cases are provided:

- Case 1: Read protection disabled after the write unprotection:
  - Erase the entire option byte area by using the OPTER bit in the Flash memory control register (FLASH_CR)
  - Program the correct RDP code 0x00A5 to unprotect the memory. This operation first forces a Mass Erase of the main Flash memory.
  - Reset the device (system reset) to reload the option bytes (and the new WRP[3:0] bytes), and to disable the write protection
- Case 2: Read protection maintained active after the write unprotection, useful for inapplication programming with a user boot loader:
  - Erase the entire option byte area by using the OPTER bit in the Flash memory control register (FLASH_CR)
  - Reset the device (system reset) to reload the option bytes (and the new WRP[3:0] bytes), and to disable the write protection.

3.5.3 Option byte block write protection

The option bytes are always read-accessible and write-protected by default. To gain write access (Program/Erase) to the option bytes, a sequence of keys (same as for lock) has to be written into the OPTKEYR. A correct sequence of keys gives write access to the option bytes and this is indicated by OPTWRE in the FLASH_CR register being set. Write access can be disabled by resetting the bit through software.

3.6 Option byte description

There are eight option bytes. They are configured by the end user depending on the application requirements. As a configuration example, the watchdog may be selected in hardware or software mode.

A 32-bit word is split up as follows in the option bytes.

Tab 3.6-1 Option byte format

31-24	23-16	15-8	7-0
complemented option byte1	Option byte 1	complemented option byte0	Option byte 0

The organization of these bytes inside the information block is as shown in Table 3.6-2. The option bytes can be read from the memory locations listed in Table 3.6-2 or from the Option byte register (FLASH_OBR).

Note: *The new programmed option bytes (user, read/write protection) are loaded after a system reset.*

Tab 3.6-2 Option byte organization

Address	31-24	23-16	15-8	7-0
0x1FFF F800	nUSER	USER	nRDP	RDP
0x1FFF F804	nData1	Data1	nData0	Data0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF F80C	nWRP3	WRP3	nWRP2	WRP2

Tab 3.6-3 Description of the option bytes

Flash address	Option bytes
0x1FFF F800	Bits [31:24] nUSER Bits [23:16] USER: User option byte (stored in FLASH_OBR[9:2]) This byte is used to configure the following features: • Select the watchdog event: Hardware or software. • Reset event when entering Stop mode. • Reset event when entering Standby mode. **Note:** *Only bits [16:18] are used, bits [23:19]: 0x1F are not used.* Bit 18: nRST_STDBY 0: Reset generated when entering Standby mode. 1: No reset generated. Bit 17: nRST_STOP 0: Reset generated when entering Stop mode 1: No reset generated Bit 16: WDG_SW 0: Hardware watchdog 1: Software watchdog Bits [15:8]: nRDP Bits [7:0]: RDP: Read protection option byte The read protection helps the user protect the software code stored in Flash memory. It is activated by setting the RDP option byte. When this option byte is programmed to a correct value (RDPRT key = 0x00A5), read access to the Flash memory is allowed. (The result of RDP level enabled/disabled is stored in FLASH_OBR[1].)
0x1FFF F804	Datax: Two bytes for user data storage. These addresses can be programmed using the option byte programming procedure. Bits [31:24]: nData1 Bits [23:16]: Data1 (stored in FLASH_OBR[25:18]) Bits [15:8]: nData0 Bits [7:0]: Data0 (stored in FLASH_OBR[17:10])

(continue)

Flash address	Option bytes
0x1FFF F808	WRPx: Flash memory write protection option bytes Bits [31:24]: nWRP1 Bits [23:16]: WRP1 (stored in FLASH_WRPR[15:8]) Bits [15:8]: nWRP0 Bits [7:0]: WRP0 (stored in FLASH_WRPR[7:0])
0x1FFF F80C	WRPx: Flash memory write protection option bytes Bits [31:24]: nWRP3 Bits [23:16]: WRP3 (stored in FLASH_WRPR[31:24]) Bits [15:8]: nWRP2 Bits [7:0]: WRP2 (stored in FLASH_WRPR[23:16]) one bit of the user option bytes WRPx is used to protect 2 pages of 1 Kbytes in main memory block. However, the bit 7 of WRP3 write protects pages 62 to 255. • 0: Write protection active • 1: Write protection not active In total, four user option bytes are used to protect the 512-Kbyte main Flash memory. WRP0: Write-protects pages 0 to 15. WRP1: Write-protects pages 16 to 31. WRP2: Write-protects pages 32 to 47. WRP3: bits 0-6 write-protect pages 48 to 61 bit 7 write-protects pages 62 to 255.

On every system reset, the option byte loader (OBL) reads the information block and stores the data into the Option byte register (FLASH_OBR) and the Write protection register (FLASH_WRPR). Each option byte also has its complement in the information block. During option loading, by verifying the option bit and its complement, it is possible to check that the loading has correctly taken place. If this is not the case, an option byte error (OPTERR) is generated. When a comparison error occurs the corresponding option byte is forced to 0xFF. The comparator is disabled when the option byte and its complement are both equal to 0xFF (Electrical Erase state).

All option bytes (but not their complements) are available to configure the product. The option registers are accessible in read mode by the CPU. See Tab 3.7-1 descriptions for more details.

3.7 Flash register

Tab 3.7-1 Flash map

Offset	Register	Reset value	Description
FLASH_BA = 0x4002_2000			
0x00	FLASH_ACR	0x0000_0030	Flash access control register
0x04	FLASH_KEYR	xxxx xxxx	FPEC key register
0x08	FLASH_OPTKEYR	xxxx xxxx	Flash OPTKEY register
0x0C	FLASH_SR	0x0000_0000	Flash status register
0x10	FLASH_CR	0x0000_0080	Flash control register
0x14	FLASH_AR	0x0000_0000	Flash address register
0x1C	FLASH_OBR	0x03FF_FFFC	Option byte register
0x20	FLASH_WRPR	0xFFFF_FFFF	Write protection register

3.7.1 Flash access control register(FLASH_ACR)

Register	Address offset	Access	Reset value	Description
FLASH_ACR	0x00	RW	0x0000_0030	Flash access control register

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

Flash access control register (FLASH_ACR)bit description

Bit	Access	Description
[31:6]	R	Reserved,must be kept at reset value
[5]	R	Reserved
[4:3]	RW	Reserved
[2:0]	RW	**LATENCY:** Latency These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time. 0 wait state, if 0 < SYSCLK ⩽ 48 MHz 1 wait state, if 48 MHz < SYSCLK ⩽ 96 MHz

3.7.2 FPEC key register(FLASH_KEYR)

Note: *These bits are all write-only and will return a 0 when read.*

Register	Address offset	Access	Reset value	Description
FLASH_KEYR	0x04	W	xxxx xxxx	FPEC key register

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

FPEC key register(FLASH_KEYR)bit description

Bit	Access	Description
[31:0]	W	**FKEYR[31:0]:** FPEC key These bits represent the keys to unlock the FPEC

3.7.3 Flash OPTKEY register(FLASH_OPTKEYR)

Note: *These bits are all write-only and will return a 0 when read.*

Register	Address offset	Access	Reset value	Description
FLASH_OPTKEYR	0x08	W	xxxx xxxx	Flash OPTKEY register

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

Flash OPTKEY register(FLASH_OPTKEYR)bit description

Bit	Access	Description
[31:0]	W	**OPTKEYR[31:0]:** Option byte key These bits represent the keys to unlock the OPTWRE.

3.7.4 Flash status register(FLASH_SR)

Register	Address offset	Access	Reset value	Description
FLASH_SR	0x0C	RW	0x0000_0000	Flash status register

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

Flash status register(FLASH_SR)bit description

Bit	Access	Description
[31:6]	R	Reserved, must be kept cleared
[5]	RW	**EOP:** End of operation Set by hardware when a Flash operation (programming / erase) is completed. Reset by writing a 1 **Note:** *EOP is asserted at the end of each successful program or erase operation*
[4]	RW	**WRPRTERR:** Write protection error Set by hardware when programming a write-protected address of the Flash memory. Reset by writing 1.
[3]	R	Reserved, must be kept cleared
[2]	RW	**PGERR:** Programming error Set by hardware when an address to be programmed contains a value different from '0xFFFF' before programming. Reset by writing 1. **Note:** *The STRT bit in the FLASH_CR register should be reset before starting a programming operation.*
[1]	R	Reserved, must be kept cleared
[0]	R	**BSY:** Busy This indicates that a Flash operation is in progress. This is set on the beginning of a Flash operation and reset when the operation finishes or when an error occurs.

3.7.5 Flash control register(FLASH_CR)

Register	Address offset	Access	Reset value	Description
FLASH_CR	0x10	RW	0x0000_0080	Flash control register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			EOPIE	Reserved	ERRIE	OPTWRE	Reserved
7	6	5	4	3	2	1	0
LOCK	STRT	OPTER	OPTPG	Reserved	MER	PER	PG

Flash control register(FLASH_CR)bit description

Bit	Access	Description
[31:13]	R	Reserved, must be kept cleared.
[12]	RW	**EOPIE:** End of operation interrupt enable This bit enables the interrupt generation when the EOP bit in the FLASH_SR register goes to 1. 0: Interrupt generation disabled 1: Interrupt generation enabled
[11]	R	Reserved, must be kept cleared.
[10]	RW	**ERRIE:** Error interrupt enable This bit enables the interrupt generation on an FPEC error (when PGERR / WRPRTERR are set in the FLASH_SR register). 0: Interrupt generation disabled 1: Interrupt generation enabled
[9]	RW	**OPTWRE:** Option bytes write enable When set, the option bytes can be programmed. This bit is set on writing the correct key sequence to the FLASH_OPTKEYR register. This bit can be reset by software
[8]	R	Reserved, must be kept cleared
[7]	RW	**LOCK:** Lock Write to 1 only. When it is set, it indicates that the FPEC and FLASH_CR are locked. This bit is reset by hardware after detecting the unlock sequence. In the event of unsuccessful unlock operation, this bit remains set until the next reset.
[6]	RW	**STRT:** Start This bit triggers an ERASE operation when set. This bit is set only by software and reset when the BSY bit is reset.
[5]	RW	**OPTER:** Option byte erase Option byte erase chosen.
[4]	RW	**OPTPG:** Option byte programming Option byte programming chosen.
[3]	R	Reserved, must be kept cleared.
[2]	RW	**MER:** Mass erase Erase of all user pages chosen.
[1]	RW	**PER:** Page erase Page Erase chosen.
[0]	RW	**PG:** Programming Flash programming chosen.

3.7.6 Flash address register(FLASH_AR)

Note: *Updated by hardware with the currently/last used address. For Page Erase operations, this should be updated by software to indicate the chosen page.*

Register	Address offset	Access	Reset value	Description
FLASH_AR	0x14	W	0x0000_0000	Flash address register

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

Flash address register(FLASH_AR)bit description

Bit	Access	Description
[31:0]	W	Flash Address Chooses the address to program when programming is selected, or a page to erase when Page Erase is selected. **Note:** *Write access to this register is blocked when the BSY bit in the FLASH_SR register is set.*

3.7.7 Option byte register(FLASH_OBR)

Note: *The reset value of this register depends on the value programmed in the option byte and the OPTERR bit reset value depends on the comparison of the option byte and its complement during the option byte loading phase*

Register	Address offset	Access	Reset value	Description
FLASH_OBR	0x1C	R	0x03FF_FFFC	Option byte register

31	30	29	28	27	26	25	24
Reserved						Data1	
23	22	21	20	19	18	17	16
Data1						Data0	
15	14	13	12	11	10	9	8
Data0						Not used	
7	6	5	4	3	2	1	0
Not used			nRST_STDBY	nRST_STOP	WDG_SW	RDPRT	OPTERR

Option byte register(FLASH_OBR)bit description

Bit	Access	Description
[31:26]	R	Reserved, must be kept cleared.
[25:18]	R	**Data1:** Data1 is stored in the Flash option byte area
[17:10]	R	**Data0:** Data0 is stored in the Flash option byte area
[9:5]	R	**USER:** User option bytes This contains the user option byte loaded by the OBL. Bits [9:5]: Not used (if these bits are written in the Flash option byte, they will be read in this register with no effect on the device.) Bit 4: nRST_STDBY Bit 3: nRST_STOP Bit 2: WDG_SW
[1]	R	**RDPRT:** Read protection When set, this indicates that the Flash memory is read-protected. **Note:** *This bit is read-only*

[0]	R	**OPTERR:** Option byte error When set, this indicates that the loaded option byte and its complement do not match. The corresponding byte and its complement are read as 0xFF in the FLASH_OBR or FLASH_WRPR register. **Note:** *This bit is read-only.*

3.7.8 Write protection register(FLASH_WRPR)

Register	Address offset	Access	Reset value	Description
FLASH_WRPR	0x20	R	0xFFFF_FFFF	Write protection register

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

Write protection register(FLASH_WRPR)bit description

Bit	Access	Description
[31:0]	R	**WRP:** Write protect This register contains the write-protection option bytes loaded by the OBL. 0: Write protection active 1: Write protection not active **Note:** *These bits are read-only.*

4 Cyclic Redundancy Check（CRC）

4.1 CRC introduction

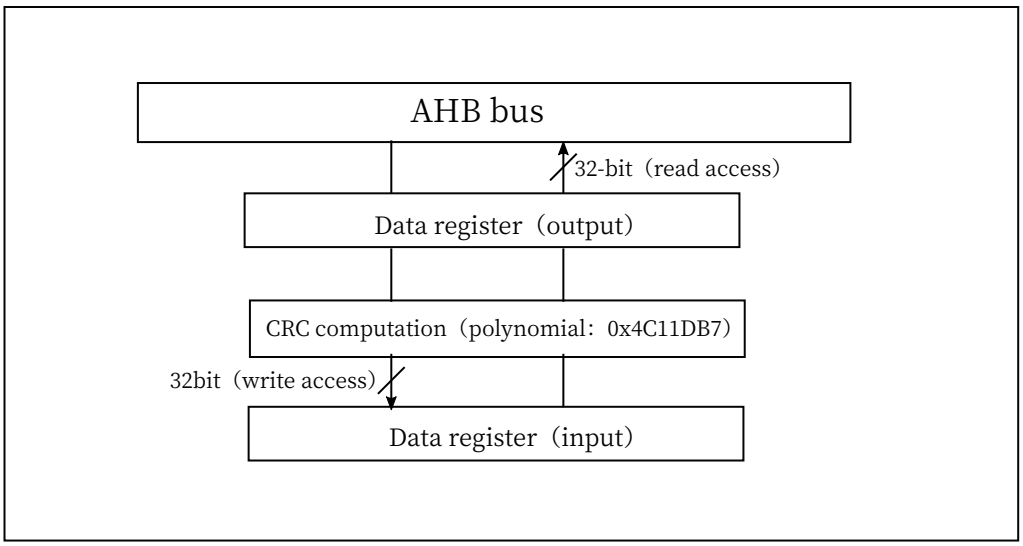
The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from a 32-bit data word and a fixed generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link- time and stored at a given memory location.

4.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^4 + X^2 + X + 1$
- Single input/output 32-bit data register
- CRC computation done in 4 AHB clock cycles (HCLK)
- General-purpose 8-bit register (can be used for temporary storage)

Fig 4.2-1 CRC block diagram



4.3 CRC functional description

The CRC calculation unit mainly consists of a single 32-bit data register, which:

- is used as an input register to enter new data in the CRC calculator (when writing into the register)
- holds the result of the previous CRC calculation (when reading the register)

Each write operation into the data register creates a combination of the previous CRC value and the new one (CRC computation is done on the whole 32-bit data word, and not byte per byte).
The write operation is stalled until the end of the CRC computation, thus allowing back-to- back write accesses or consecutive write and read accesses.

The CRC calculator can be reset to 0xFFFF FFFF with the RESET control bit in the CRC_CR register. This operation does not affect the contents of the CRC_IDR register.

4.4 CRC Registers

Tab 4.4-1 CRC registers map

offset	Register	Reset value	Description
CRC: CRC_BA = 0x4002_3000			
0x00	CRC_DR	0xFFFF_FFFF	CRC Data register
0x04	CRC_IDR	0xFFFF_FFFF	CRC Independent data register
0x08	CRC_CR	0xFFFF_FFFF	CRC Control register

4.4.1 CRC Data register(CRC_DR)

Register	Address offset	Access	Reset value	Description
CRC_DR	0x00	RW	0xFFFF_FFFF	CRC Data register

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

CRC Data register(CRC_DR)bit description

Bit	Access	Description
[31:0]	RW	**DR[31:0]:** Data register bits Used as an input register when writing new data into the CRC calculator. Holds the previous CRC calculation result when it is read

4.4.2 CRC Independent data register(CRC_IDR)

Register	Address offset	Access	Reset value	Description
CRC_IDR	0x04	RW	0xFFFF_FFFF	CRC Independent data register

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

CRC Independent data register(CRC_IDR)bit description

Bit	Access	Description
[31:8]	R	Reserved, must be kept at reset value.

[7:0]	RW	**IDR[7:0]:** General-purpose 8-bit data register bits Can be used as a temporary storage location for one byte. This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register.

Note: *This register does not participate in CRC calculations and can hold any data.*

4.4.3 CRC Control register(CRC_CR)

Register	Address offset	Access	Reset value	Description
CRC_CR	0x08	W	0xFFFF_FFFF	CRC Control register

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

CRC Control register(CRC_CR)bit description

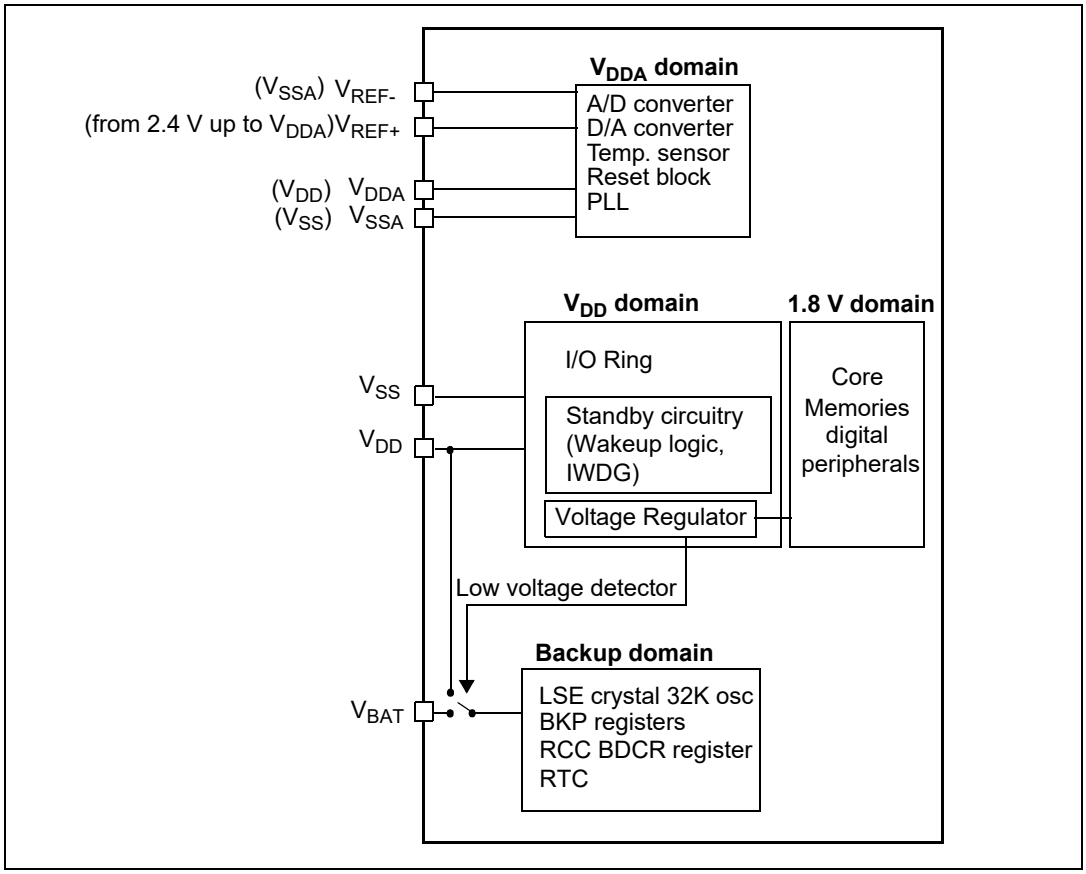
Bit	Access	Description
[31:1]	-	Reserved, must be kept at reset value.
[0]	W	**RESET:** RESET bit Resets the CRC calculation unit and sets the data register to 0xFFFF_FFFF. This bit can only be set, it is automatically cleared by hardware.

5 Power control(PWR)

5.1 Power supplies

MG32F157xx requires a 2.0-to-3.6 V operating voltage supply (VDD). An embedded regulator is used to supply the internal 1.8 V digital power. The real-time clock (RTC) and backup registers can be powered from the VBAT voltage when the main V DD supply is powered off.

Fig 5.1-1 Power supply overview



1. V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.

5.1.1 Independent A/D and D/A converter supply and reference voltage

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

- The ADC and DAC voltage supply input is available on a separate VDDA pin.
- An isolated supply ground connection is provided on pin VSSA .
- VREF+/ VREF- reference voltage pin

When available (according to package), VREF- must be tied to VSSA .

On 100-pin:
To ensure a better accuracy on low-voltage inputs and outputs, the user can connect a separate external reference voltage on VREF+ . VREF+ is the highest voltage, represented by the full scale value,

for an analog input (ADC) or output (DAC) signal. The voltage on VREF+ can range from 2.4 V to VDDA.

On 64-pin packages:
The VREF+ and VREF- pins are not available, they are internally connected to the ADC voltage supply (VDDA) and ground (VSSA).

5.1.2 Battery backup domain

To retain the content of the Backup registers and supply the RTC function when VDD is turned off, VBAT pin can be connected to an optional standby voltage supplied by a battery or by another source. The VBAT pins power the RTC unit , the LSE oscillator and the PC13 to PC15 IOs, allowing the RTC to operate even when the main power supply (VDD) is turned off. The switch of the VBAT power supply is controlled by the power-off reset function embedded in the reset module.
Note:

- *During tRSTTEMPO (temporization at VDD startup) or after a PDR is detected, the power switch between VBAT and VDD remains connected to VBAT .*
- *During the startup phase, if VDD is established in less than tRSTTEMPO (Refer to the datasheet for the value of t RSTTEMPO) and VDD > VBAT + 0.6 V, a current may be injected into V BAT through an 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 external batteries are not used in the application, it is recommended to use a 100 nF external ceramic decoupling capacitor to connect the VBAT external to the VDD.
When the backup domain is provided by a VDD (to which the analog switch is connected), the following functions are available:

- PC14 and PC15 can be used as either GPIO or LSE pins
- PC13 can be used as GPIO, TAMPER pin, RTC Calibration Clock, RTC Alarm or second output ***BKP_RTCCR***

Note: *Due to the fact that the switch only sinks a limited amount of current (3 mA), the use of GPIOs PC13 to PC15 in output mode is restricted: the speed has to be limited to 2 MHz with a maximum load of 30 pF and these IOs must not be used as a current source (e.g. to drive a LED).* When the backup domain is supplied by VBAT (analog switch connected to VBAT), the following functions are available:

- PC14 and PC15 can be used as LSE pins only
- PC13 can be used as TAMPER pin

5.1.3 Voltage regulator(LDO)

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

- In Run mode, the regulator supplies full power to the 1.8 V domain (core, memories and digital peripherals).
- In Stop mode the regulator supplies low-power to the 1.8 V domain, preserving contents of registers and SRAM
- In Standby Mode, the regulator is powered off. The contents of the registers and SRAM are lost

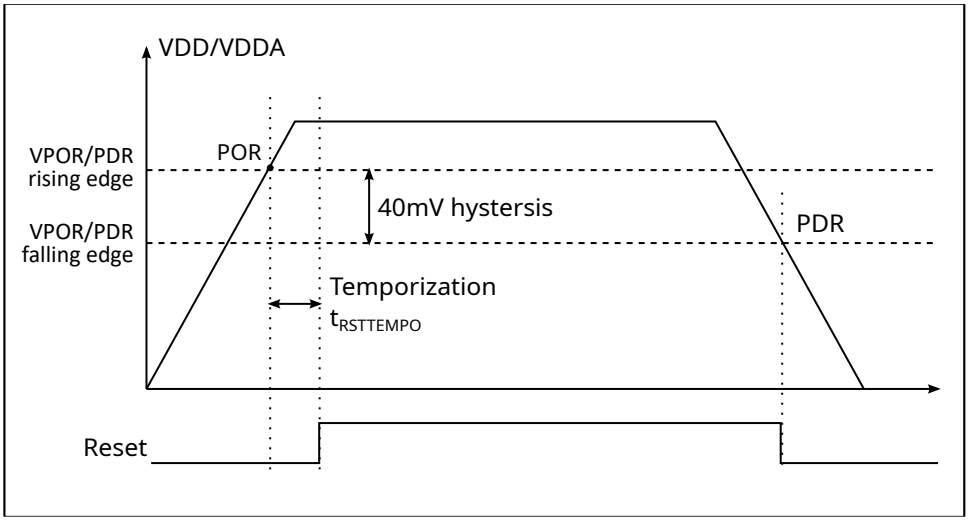
except for the Standby circuitry and the Backup Domain.

5.2 Power supply supervisor

5.2.1 Power on reset (POR)/power down reset (PDR)

MG32F157xx has an integrated POR/PDR circuitry that allows proper operation starting from/down to 2 V . MG32F157xx remains in Reset mode when VDD /VDDA is below a specified threshold, V_{POR} /V_{PDR}, without the need for an external reset circuit.

Fig 5.2-1 Power on reset/power down reset waveform



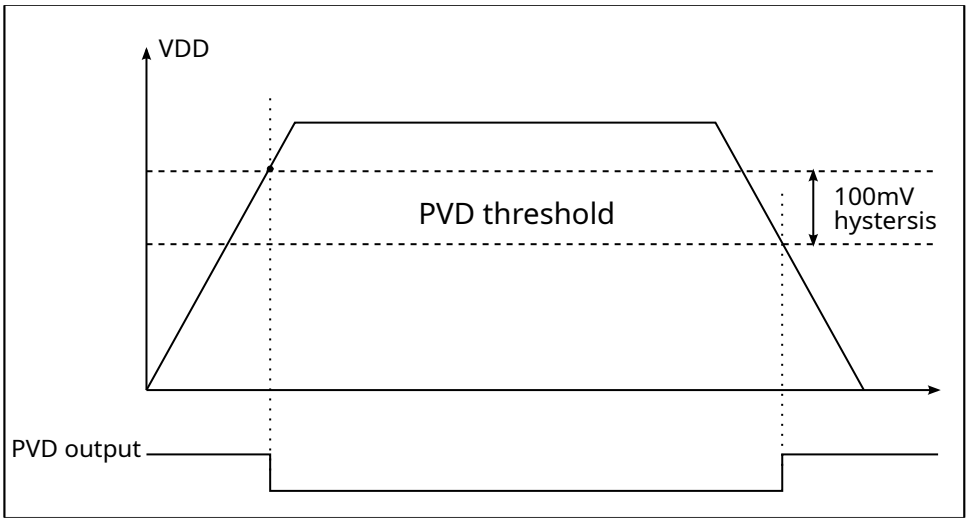
5.2.2 Programmable voltage detector(PVD)

The PVD can be used to monitor the VDD /VDDA power supply by comparing it to a threshold selected by the PLS[2:0] bits in the Power control register(PWR_CR)

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the Power control/status register (PWR_CSR), to indicate if VDD /VDDA is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when VDD /VDDA drops below the PVD threshold and/or when VDD /VDDA rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example the service routine could perform emergency shutdown tasks.

Fig 5.2-2 PVD thresholds



5.3 Low-power modes

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

three low-power modes:

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

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

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

Tab 5.3-1 Low-power mode summary

Mode name	Entry	Wakeup	Effect on 1.8V domain clocks	Effect on VDD domain clocks	Voltage regulator
Sleep (Sleep now or Sleep-on-exit)	WFI	Any interrupt	CPU clock OFF, no effect on digital clocks or analog clock sources	None	ON
	WFE	Wakeup event			
Stop	PDDS and LPDS bits + SLEEPDEEP bit + WFI/WFE	Any EXTI line(configured in the EXTI registers)	1.8v domain clocks OFF	HSI and HSE oscillators OFF	ON or in low-power mode（PWR_CR）
Standby	PDDS bit + SLEEPDEEP bit +WFI/WFE	WKUP pin rising edge or falling edge（PWR_CR control of STBY_REDGE） 、RTC alarm、reset in NRST pin、IWDG reset			OFF

5.3.1 Slowing down system clocks

In Run mode the speed of the system clocks (SYSCLK, HCLK, PCLK1, PCLK2) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down peripherals before entering Sleep mode.For more details refer to ***RCC***

5.3.2 Peripheral clock gating

In Run mode, the HCLK and PCLKx for individual peripherals and memories can be stopped at any time to reduce power consumption.To further reduce power consumption in Sleep mode the peripheral clocks can be disabled prior to executing the WFI or WFE instructions.

Peripheral clock gating is controlled by the ***RCC_AHBENR, RCC_APB1ENR, RCC_APB2ENR***

5.3.3 Sleep mode

Entering Sleep mode The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex™-M3 System Control register:

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR

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

Exiting Sleep mode

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

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

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex™-M3 System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set. This mode offers the lowest wakeup time as no time is wasted in interrupt entry/exit.

Tab 5.3-2 Sleep mode entry and exit

SLEEP-NOW	
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: -SLEEPDEEP = 0 -SLEEPONEXTI = 0 Refer to the Cortex-M3 System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to Section:10 If WFE was used for entry Wakeup event: Refer to Section:10
Wakeup latency	None
SLEEP-ON-EXTI	
Mode entry	WFI (wait for interrupt) while: -SLEEPDEEP = 0 -SLEEPONEXTI = 1 Refer to the Cortex-M3 System Control register.
Mode exit	Interrupt: refer to Section:10。
Wakeup latency	None

5.3.4 Stop mode

The Stop mode is based on the Cortex™-M3 deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode. In Stop mode, all clocks in the 1.2 V domain are stopped, the PLL, the HSI and the HSE RC oscillators are disabled. SRAM and register contents are preserved.

In the Stop mode, all I/O pins keep the same state as in the Run mode。

Entering Stop mode

To further reduce power consumption in Stop mode, the internal voltage regulator can be put in low-power mode. This is configured by the LPDS bit of the Power control register(PWR_CR).

If Flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.
If an access to the APB domain is ongoing, The Stop mode entry is delayed until the APB access is finished.

In Stop mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset（***IWDG***）
- Real-time clock (RTC): this is configured by the ***RCC_BDCR***
- Internal RC oscillator (LSI RC):
  1. ***RCC_CSR_LSION***
  2. ***PWR_STBY_LSION***
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the ***RCC_BDCR***.

The ADC or DAC can also consume power during the Stop mode, unless they are disabled before entering it. To disable them, the ADON bit in the ADC_CR2 register and the ENx bit in the DAC_CR register must both be written to 0.

Note: *If the application needs to disable the external clock before entering Stop mode, the HSEON bit must first be disabled and the system clock switched to HSI. Otherwise, if the HSEON bit remains enabled and the external clock (external oscillator) is removed when entering Stop mode, the clock security system (CSS) feature must be enabled to detect any external oscillator failure and avoid a malfunction behavior when entering stop mode.*

Exiting Stop mode

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

When the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop mode. By keeping the internal regulator ON during Stop mode, the consumption is higher although the startup time is reduced.

Tab 5.3-3 Stop mode entry and exit

Stop mode	
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: • Set SLEEPDEEP bit in Cortex-M3 System Control register • The PDDS bit of the PWR_CR configuration register is 0 • Clear WUF bit in Power Control register (PWR_CR) • Select the voltage regulator mode by configuring LPDS bit in PWR_CR **Note:** *To enter Stop mode, all EXTI Line pending bits (in Pending register (EXTI_PR)), all peripheral interrupt pending bits, and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.*
Mode exit	(1)If WFI was used for entry: Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC).Refer to***NVIC*** (2)If WFE was used for entry: Any EXTI Line configured in event mode. Refer to***NVIC***
Wakeup latency	HSI RC wakeup time + regulator wakeup time from Low-power mode

5.3.5 Standby mode

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

Entering Standby mode

In Standby mode, the following features can be selected：

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

Exiting Standby mode

The microcontroller exits the Standby mode when an external reset (NRST pin), an IWDG reset, rising/falling edge on the WKUP pin（selected by STBY_WKUP_EDGE of the PWR_STBY）or the rising edge of an RTC alarm occurs . All registers are reset after wakeup from Standby except for PWR_CSR and PWR_STBY. After waking up from Standby mode, program execution restarts in the same way as after a Reset (boot pins sampling, vector reset is fetched, etc.). The SBF status flag in the Power control/status register (PWR_CSR) indicates that the system is woken up from standby mode.

Tab 5.3-4 Standby mode entry and exit

Standby mode	
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: –Set SLEEPDEEP in Cortex-M3 System Control register –Set PDDS bit in Power Control register (PWR_CR) –Clear WUF bit in Power Control/Status register (PWR_CSR) –No interrupt (for WFI) or event (for WFI) is pending
Mode exit	- WKUP pin rising edge - RTC alarm event’s rising edge - External reset on NRST pin - IWDG reset
Wakeup latency	Reset phase

I/O states in Standby mode

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

- Reset pad (still available)
- TAMPER pins configured for intrusion prevention
- WKUP pin enabled

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex™-M3 core is no longer clocked. However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged when using the low-power modes

5.3.6 Auto-wakeup from low-power mode(AWU)

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

- Low-power 32.768 kHz external crystal oscillator（LSE OSC）:
  This clock source provides a precise time base with very low-power consumption (less than 0.5μA added consumption in typical conditions)
- Low-power internal RC Oscillator（LSI RC）:
  This clock source has the advantage of saving the cost of the 32.768 kHz crystal. But the internal RC oscillator will add a little power consumption

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

- Configure the EXTI Line 17 to be sensitive to rising edge
- Configure the RTC to generate the RTC alarm

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

5.4 Power control register

Tab 5.4-1 PWR register mapping table

Address offset	Registers	Reset value	Description
PWR Base address: PWR_BA = 0x4000_7000			
0x00	PWR_CR	0x0000_0000	PWR control register
0x04	PWR_CSR	0x0000_0000	PWR control/status register
0x08	PWR_STBY	0x0000_0000	PWR standby control register

5.4.1 PWR control register(PWR_CR)

Register	Address offset	Access	Reset value	Description
PWR_CR	0x00	RW	0x0000_0000	PWR control register

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

PWR control register(PWR_CR)bit description

Bit	Access	Description
[31:9]	R	Reserved
[8]	RW	**DBP:** Disable backup domain write protection In reset state, the RTC and backup registers are protected against parasitic write access. This bit must be set to enable write access to these registers. 0: Access to RTC and Backup registers disabled 1: Access to RTC and Backup registers enabled **Note:** *If the HSE divided by 128 is used as the RTC clock, this bit must remain set to 1.*
[7:5]	RW	**PLS[2:0]:** PVD level selection To select the voltage threshold detected by the Power Voltage Detector 000: 2.2V 001: 2.3V 010: 2.4V 011: 2.5V 100: 2.6V 101: 2.7V 110: 2.8V 111: 2.9V
[4]	RW	**PVDE:** Power voltage detector enable. This bit is set and cleared by software. 0: PVD disabled 1: PVD enabled
[3]	RC_W1	**CSBF:** Clear standby flag. This bit is always read as 0. 0: No effect 1: Clear the SBF Standby Flag (write).

[2]	RC_W1	**CWUF:** Clear wakeup flag This bit is always read as 0 0: No effect 1: Clear the WUF Wakeup Flag after 2 System clock cycles (write).
[1]	RW	**PDDS:** Power down deepsleep. This bit is set and cleared by software. It works together with the LPDS bit 0: Enter Stop mode when the CPU enters Deepsleep. The regulator status depends on the LPDS bit. 1: Enter Standby mode when the CPU enters Deepsleep
[0]	RW	**LPDS:** Low-power deepsleep. This bit is set and cleared by software. It works together with the PDDS bit.(PDDS=0) 0: Voltage regulator on during Stop mode 1: Voltage regulator in low-power mode during Stop mode

5.4.2 PWR control/status register(PWR_CSR)

Note: *Does not clear when awakened from standby mode*

Register	Address offset	Access	Reset value	Description
PWR_CSR	0x04	R	0x0000_0000	PWR control/status register

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

PWR control/status register(PWR_CSR)bit description

Bit	Access	Description
[31:9]	R	Reserved
[8]	RW	**EWUP:** Enable WKUP pin This bit is set and cleared by software 0: WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wakeup the device from Standby mode 1: WKUP pin is used to wake up the CPU from standby mode (The rising or falling edge of the WKUP pin wakes the system up from standby mode, STBY_WKUP_EDGE of PWR_STBY selects rising edge or falling edge) **Note:** *This bit is reset by a system Reset*
[7:3]	R	Reserved

[2]	R	**PVDO:** PVD output This bit is set and cleared by hardware. It is valid only if PVD is enabled by the PVDE bit. 0: VDD/VDDA is higher than the PVD threshold selected with the PLS[2:0] bits 1: VDD/VDDA is lower than the PVD threshold selected with the PLS[2:0] bits. **Note:** *The PVD is stopped by Standby mode. For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set*
[1]	R	**SBF:** Standby flag This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CSBF bit in the PWR_CR 0: Device has not been in Standby mode 1: Device has been in Standby mode
[0]	R	**WUF:** Wakeup flag This bit is set by hardware and cleared by hardware, by a system reset or by setting the CWUF bit in the Power control register (PWR_CR) 0: No wakeup event occurred 1: A wakeup event was received from the WKUP pin or from the RTC alarm **Note:** *An additional wakeup event is detected if the WKUP pin is enabled (by setting the EWUP bit)* : *(1)PWR_STBY_WKUP_EDGE = 0, The WKUP pin is already configured with EWUP at low power.* *(2)PWR_STBY_WKUP_EDGE = 1, The WKUP pin is already configured with EWUP at high power。*

5.4.3 PWR standby control register(PWR_STBY)

Note: *Does not clear when awakened from standby mode*

Register	Address offset	Access	Reset value	Description
PWR_STBY	0x08	R	0x0000_0000	PWR standby control register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
EWUP							
7	6	5	4	3	2	1	0
EWUP	Reserved0	STBY_WKUP_	STBY_LSION	Reserved			Reserved0

PWR standby control register(PWR_STBY)bit description

Bit	Access	Description
[31:16]	R	[31:16]must be set to 0x5a69, [15:0] configuration is valid.

[15:7]	R	**EWUP:** WKUP pin enabled This bit is set and cleared by the software 0: The WKUP pin is general I/O. Events on the WKUP pin do not wake up the device from standby mode 1: The WKUP pin is used to wake up from standby mode and force entry into the input drop-down configuration, (the rising edge of the WKUP pin wakes up the system from standby mode) **Note:** *The bit is reset by the system reset*
[6]	RW	**Reserved:** The value must be set to 0
[5]	R	**STBY_WKUP_EDGE:** PA0 wake up control: 0:Falling edge awakening 1:Rising edge awakening
[4]	RW	**STBY_LSION:** LSI control: 0:OFF 1:ON
[3:1]	R	Reserved
[0]	RW	**Reserved:** The value must be set to 0

6 Backup registers(BKP)

6.1 BKP introduction

The backup registers are forty two 16-bit registers for storing 84 bytes of user application data.They are implemented in the backup domain that remains powered on by V BAT when the VDD power is switched off. They are not reset when the device wakes up from Standby mode or by a system reset or power reset.In addition, the BKP control registers are used to manage the Tamper detection feature and RTC calibration.

After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access. To enable access to the Backup registers and the RTC, proceed as follows:

- enable the power and backup interface clocks by setting the PWREN and BKPEN bits in the RCC_APB1ENR register
- set the DBP bit the Power Control Register (PWR_CR) to enable access to the Backup registers and RTC.

6.2 BKP main features

- 84-byte data registers
- Status/control register for managing tamper detection with interrupt capability
- Calibration register for storing the RTC calibration value
- Possibility to output the RTC Calibration Clock, RTC Alarm pulse or Second pulse on TAMPER pin PC13 (when this pin is not used for tamper detection)

6.3 BKP functional description

6.3.1 Tamper detection

The TAMPER pin generates a Tamper detection event when the pin changes from 0 to 1 or from 1 to 0 depending on the TPAL bit in the Backup control register (BKP_CR). A tamper detection event resets all data backup registers. However to avoid losing Tamper events, the signal used for edge detection is logically ANDed with the Tamper enable in order to detect a Tamper event in case it occurs before the TAMPER pin is enabled

- TPAL = 0 : If the TAMPER pin is already high before it is enabled , an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no rising edge on the TAMPER pin after TPE was set)
- TPAL = 1 ： If the TAMPER pin is already low before it is enabled (by setting the TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no falling edge on the TAMPER pin after TPE was set)

By setting the TPIE bit in the BKP_CSR register, an interrupt is generated when a Tamper detection event occurs.

After a Tamper event has been detected and cleared, the TAMPER pin should be disabled and then re-enabled with TPE before writing to the backup data registers (BKP_DRx) again. This prevents software from writing to the backup data registers (BKP_DRx), while the TAMPER pin value still indicates a Tamper detection. This is equivalent to a level detection on the TAMPER pin.

Note: *Tamper detection is still active when VDD power is switched off. To avoid unwanted resetting of the data backup registers, the TAMPER pin should be externally tied to the correct level.*

6.4 RTC calibration

For measurement purposes, the RTC clock with a frequency divided by 64 can be output on the TAMPER pin. This is enabled by setting the CCO bit in the RTC clock calibration register (BKP_RTCCR). The clock can be slowed down by up to 121 ppm by configuring CAL[6:0] bits.

6.5 BKP registers

Tab 6.5-1 BKP register map

Address offset	Register	Reset value	Description
BKP base address: BKP_BA = 0x4000_6C00			
0x04 ~ 0x28 0x40 ~ 0xBC	BKP_DRx	0x0000_0000	Backup data register x (x = 1~42)
0x2C	BKP_RTCCR	0x0000_0000	RTC clock calibration register
0x30	BKP_CR	0x0000_0000	Backup control register
0x34	BKP_CSR	0x0000_0000	Backup control/status register

6.5.1 Backup data register x (BKP_DRx)(x=1..42)

Register	Address offset	Access	Reset value	Description
BKP_DRx	0x04 - 0x28, 0x40 - 0xBC	RW	0x0000_0000	Backup data register x (x = 1..42)

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

Backup data register x (BKP_DRx)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**Data[15:0]:** Backup data These bits can be written with user data **Note:** *The BKP_DRx registers are not reset by a System reset or Power reset or when the device wakes up from Standby mode. They are reset by a Backup Domain reset or by a TAMPER pin event (if the TAMPER pin function is activated).*

6.5.2 RTC clock calibration register (BKP_RTCCR)

Register	Address offset	Access	Reset value	Description
BKP_RTCCR	0x2C	RW	0x0000_0000	RTC clock calibration register

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

RTC clock calibration register (BKP_RTCCR)bit description

Bit	Access	Description
[31:10]	-	Reserved,must be kept at reset value.
[9]	RW	**ASOS:** Alarm or second output selection When the ASOE bit is set, the ASOS bit can be used to select whether the signal output on the TAMPER pin is the RTC Second pulse signal or the Alarm pulse signal: 0: RTC Alarm pulse output selected 1: RTC Second pulse output selected **Note:** *This bit is reset only by a Backup domain reset.*
[8]	RW	**ASOE:** Alarm or second output enable Setting this bit outputs either the RTC Alarm pulse signal or the Second pulse signal on the TAMPER pin depending on the ASOS bit. The output pulse duration is one RTC clock period. The TAMPER pin must not be enabled while the ASOE bit is set. **Note:** *This bit is reset only by a Backup domain reset*
[7]	RW	**CCO:** Calibration clock output 0: No effect 1: Setting this bit outputs the RTC clock with a frequency divided by 64 on the TAMPER pin. The TAMPER pin must not be enabled while the CCO bit is set in order to avoid unwanted Tamper detection **Note:** *This bit is reset when the VDD supply is powered off*
[6:0]	RW	**CAL6:0:** Calibration value This value indicates the number of clock pulses that will be ignored every 2^{20} clock pulses. This allows the calibration of the RTC, slowing down the clock by steps of 1000000/2^{20} PPM The clock of the RTC can be slowed down from 0 to 121PPM.

6.5.3 Backup control register(BKP_CR)

Register	Address offset	Access	Reset value	Description
BKP_CR	0x30	RW	0x0000_0000	Backup control register

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

Backup control register(BKP_CR)bit description

Bit	Access	Description
[31:2]	R	Reserved
[1]	RW	**TPAL:** TAMPER pin active level 0: A high level on the TAMPER pin resets all data backup registers (if TPE bit is set) 1: A low level on the TAMPER pin resets all data backup registers (if TPE bit is set).
[0]	RW	**TPE:** TAMPER pin enable 0: The TAMPER pin is free for general purpose I/O 1: Tamper alternate I/O function is activated

6.5.4 Backup control/status register(BKP_CSR)

Register	Address offset	Access	Reset value	Description
BKP_CSR	0x34	RW	0x0000_0000	Backup control/status register

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

Backup control/status register(BKP_CSR)bit description

Bit	Access	Description
[31:10]	-	Reserved, must be kept at reset value.
[9]	R	**TIF:** Tamper interrupt flag This bit is set by hardware when a Tamper event is detected and the TPIE bit is set. It is cleared by writing 1 to the CTI bit (also clears the interrupt). It is also cleared if the TPIE bit is reset. 0: No Tamper interrupt 1: A Tamper interrupt occurred **Note:** *This bit is reset only by a system reset and wakeup from Standby mode.*

[8]	R	**TEF:** Tamper event flag This bit is set by hardware when a Tamper event is detected. It is cleared by writing 1 to the CTE bit. 0: No Tamper event 1: A Tamper event occurred **Note:** *A Tamper event resets all the BKP_DRx registers. They are held in reset as long as the TEF bit is set. If a write to the BKP_DRx registers is performed while this bit is set, the value will not be stored.*
[7:3]	R	Reserved,must be kept at reset value
[2]	RW	**TPIE:** TAMPER pin interrupt enable 0: Tamper interrupt disabled 1: Tamper interrupt enabled (the TPE bit must also be set in the BKP_CR register) **Note:** *A Tamper interrupt does not wake up the core from low-power modes.* *This bit is reset only by a system reset and wakeup from Standby mode.*
[1]	W	**CTI:** Clear tamper interrupt This bit is write only, and is always read as 0. 0: No effect 1: Clear the Tamper interrupt and the TIF Tamper interrupt flag.
[0]	W	**CTE:** Clear tamper event This bit is write only, and is always read as 0. 0: No effect 1: Reset the TEF Tamper event flag (and the Tamper detector)

7 Reset and clock controller(RCC)

7.1 Reset function description

There are three types of reset, defined as system reset, power reset and backup domain reset.

7.2 Reset

7.2.1 System reset

A system reset sets all registers to their reset values except the reset flags in the clock controller CSR register and the registers in the Backup domain.
A system reset is generated when one of the following events occurs:

- A low level on the NRST pin (external reset)
- Window watchdog end of count condition (WWDG reset)
- Independent watchdog end of count condition (IWDG reset)
- A software reset (SW reset)
- Low-power management reset

The reset source can be identified by checking the reset flags in the Control/Status register(RCC_CSR)

Software reset

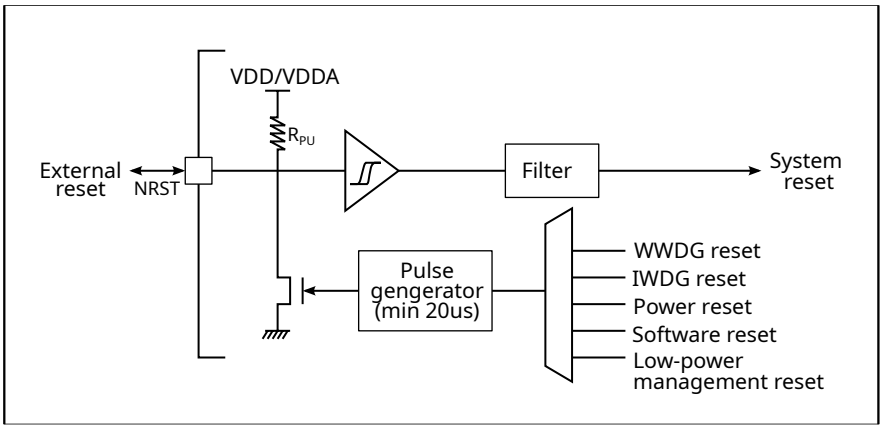
Cortex™-M3 can perform a software reset by setting the SYSRESETREQ bit of the internal register to 1.

Low power reset

There are two ways to generate a low-power reset:

- Reset generated when entering Standby mode: By setting the nRST_STDBY bit in the flash user byte option, the standby mode sequence can be reset after successful execution, and the standby mode will exit when reset
- Reset when entering Stop mode: By setting the nRST_STOP bit in the flash reset user byte option, the stop mode can be reset after successful execution of the stop mode sequence, and the stop mode will exit when reset

Fig 7.2-1 Reset circuit block diagram



7.2.2 Power reset

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

- Power-on/power-down reset (POR/PDR reset)
- When exiting Standby mode

A power reset sets all registers to their reset values except the Backup domain

These sources act on the NRST pin and it is always kept low during the delay phase. The RESET service routine vector is fixed at address 0x0000_0004 in the memory map.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each reset source (external or internal reset). In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

7.2.3 Backup domain reset

The backup domain has two specific resets that affect only the backup domain :

- Software reset, triggered by setting the BDRST bit in the Backup domain control register (RCC_BDCR).
- VDD or VBAT power on, if both supplies have previously been powered off.

7.3 Clocks

Three different clock sources can be used to drive the system clock (SYSCLK):

1. HSI oscillator clock
2. HSE oscillator clock
3. PLL clock

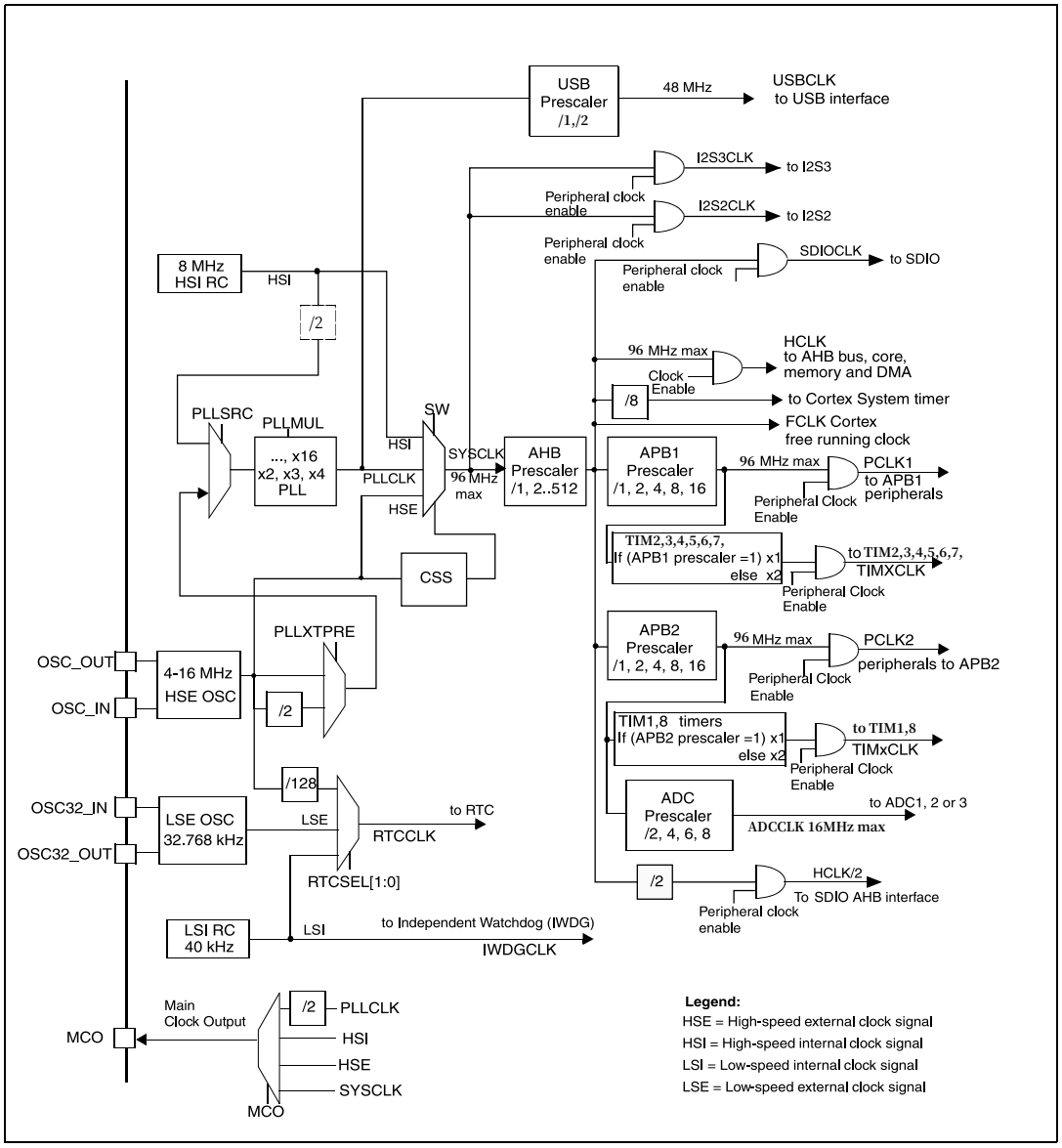
The devices have the following two secondary clock sources:

1. 40 kHz low speed internal RC (LSI RC), which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop/Standby mode.

2. 32.768 kHz low speed external crystal (LSE crystal), which optionally drives the real-time clock (RTCCLK)

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Fig 7.3-1 Clock tree



Note: *When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 64 MHz.*

Several prescalers allow the configuration of the AHB frequency, the high speed APB (APB2) and the low speed APB (APB1) domains. The maximum frequency of the AHB and the APB2 domains is 96MHz. The maximum allowed frequency of the APB1 domain is 96MHz. The SDIO AHB interface is clocked with a fixed frequency equal to HCLK/2.

The RCC feeds the Cortex System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the Cortex clock (HCLK), configurable in the

SysTick Control and Status Register. The ADCs are clocked by the clock of the High Speed domain (APB2) divided by 2, 4, 6 or 8.

The Flash memory programming interface clock (FLITFCLK) is always the HSI clock.

The timer clock frequencies are automatically fixed by hardware. There are two cases:

1. if the APB prescaler is 1, the timer clock frequencies are set to the same frequency as that of the APB domain to which the timers are connected.
2. otherwise, they are set to twice (×2) the frequency of the APB domain to which the timers are connected.

FCLK is the running clock source for Cortex™-M3

7.3.1 HSE clock

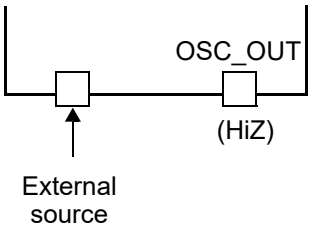
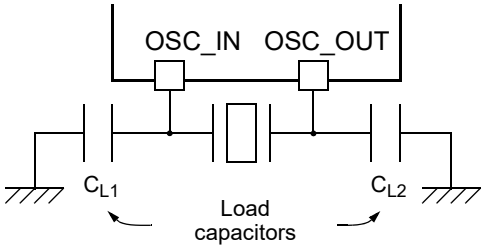
The high speed external clock signal (HSE) can be generated from two possible clock sources:

- HSE external crystal/ceramic resonator
- HSE user external clock

The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator

1. HSE external crystal/ceramic resonator
   The 4 to 16 MHz external oscillator has the advantage of producing a very accurate rate on the main clock. The HSERDY flag in the Clock control register (RCC_CR) indicates if the high-speed external oscillator is stable or not.After each boot, the hardware will wait until HSERDY is high before using the clock
   An interrupt can be generated if enabled in the Clock interrupt register (RCC_CIR).
   The HSE Crystal can be switched on and off using the HSEON bit in the Clock control register (RCC_CR).
2. HSE user external clock
   In this mode, an external clock source must be provided. It can have a frequency of up to 25 MHz.
   Select this mode by setting the HSEBYP and HSEON bits in the Clock control register (RCC_CR).
   The external clock signal (square, sine or triangle) with 50% duty cycle has to drive the OSC_IN pin while the OSC_OUT pin should be left hi-z.

Fig 7.3-2 HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	

7.3.2 HSI clock

The HSI clock signal is generated from an internal 8 MHz RC Oscillator and can be used directly as a system clock or divided by 2 to be used as PLL input.
The HSI RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1% accuracy at T_A = 25 °C
After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the Clock control register (RCC_CR).
If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. You can trim the HSI frequency in the application using the HSITRIM[4:0] bits in the Clock control register (RCC_CR).
The HSIRDY flag in the Clock control register (RCC_CR) indicates if the HSI RC is stable or not. The hardware does not use the clock until HSIRDY is high after each boot.
The HSI RC can be switched on and off using the HSION bit in the Clock control register(RCC_CR). The HSI signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails

7.3.3 PLL

The internal PLL can be used to multiply the HSI RC output or HSE crystal output clock frequency. Refer to Clock control register (RCC_CR).
The PLL configuration (selection of HSI oscillator divided by 2 or HSE oscillator for PLL input clock, and multiplication factor) must be done before enabling the PLL. Once the PLL enabled, these parameters cannot be changed.
An interrupt can be generated when the PLL is ready if enabled in the Clock interrupt register (RCC_CIR).
If the USB interface is used in the application, the PLL must be programmed to output 48 or 96 MHz. This is needed to provide a 48 MHz USBCLK.

7.3.4 LSE clock

The source of the LSE clock is the same as that of the HSE clock, with two input sources:

1. 32.768 kHz Low Speed External crystal or ceramic resonator (LSE)
   It has the advantage providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions. The LSE crystal is switched on and off using the LSEON bit in Backup domain control register (RCC_BDCR).The LSERDY flag in the Backup domain control register (RCC_BDCR) indicates if the LSE crystal is stable or not.After each boot, the hardware will wait until the LSERDY is raised before using the clock. An interrupt can be generated if enabled in the Clock interrupt register (RCC_CIR).
2. External source (LSE bypass)
   In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. You select this mode by setting the LSEBYP and LSEON bits in the Backup domain control register (RCC_BDCR). The external clock signal (square, sine or triangle) with 50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be kept in a high resistance state.

7.3.5 LSI clock

The LSI RC acts as an low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and Auto-wakeup unit (AWU). The clock frequency is around 40 kHz (between 30 kHz and 60 kHz)

The LSI RC can be switched on and off using the LSION bit in the Control/status register(RCC_CSR). The LSIRDY flag in the Control/status register (RCC_CSR) indicates if the low-speed internal(LSI) oscillator is stable or not.At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the Clock interrupt register(RCC_CIR).

LSI calibration

The frequency dispersion of the Low Speed Internal RC (LSI) oscillator can be calibrated to have accurate RTC time base and/or IWDG timeout (when LSI is used as clock source for these peripherals) with an acceptable accuracy.

This calibration is performed by measuring the LSI clock frequency with respect to TIM5 input clock

(TIM5CLK). According to this measurement done at the precision of the HSE oscillator, the software can adjust the programmable 20-bit prescaler of the RTC to get an accurate time base or can compute accurate IWDG timeout.

Use the following procedure to calibrate the LSI:

1. Enable TIM5 timer and configure channel4 in input capture mode
2. Set the TIM5CH4_IREMAP bit in the AFIO_MAPR register to connect the LSI clock internally to TIM5 channel4 input capture for calibration purpose.
3. Measure the frequency of LSI clock using the TIM5 Capture/compare 4 event or interrupt。
4. Use the measured LSI frequency to update the 20-bit prescaler of the RTC depending on the desired time base and/or to compute the IWDG timeout.

7.3.6 System clock (SYSCLK) selection

After a system reset, the HSI oscillator is selected as system clock. When a clock source is used directly or through the PLL as system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch will occur when the clock source will be ready. Status bits in the Clock control register (RCC_CR) indicate which clock(s) is (are) ready and which clock is currently used as system clock.

7.3.7 Clock security system(CSS)

Clock Security System can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped. If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers (TIM1 and TIM8) and an interrupt is generated to inform the software about the failure (Clock Security System Interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex™-M3 NMI (Non-Maskable Interrupt) exception vector.

Note: *Once the CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and an NMI is automatically generated. The NMI will be executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the Clock interrupt register (RCC_CIR).*

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock, and the PLL clock is used as system clock), a detected failure causes a switch of the system clock to the HSI oscillator and the disabling of the HSE oscillator. If the HSE clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

7.3.8 RTC clock

The RTCCLK clock source can be either the HSE/128, LSE or LSI clocks. This is selected by programming the RTCSEL[1:0] bits in the Backup domain control register (RCC_BDCR). This selection cannot be modified without resetting the Backup domain.

The LSE clock is in the Backup domain, whereas the HSE and LSI clocks are not Consequently:

- If LSE is selected as RTC clock:The RTC continues to work even if the VDD supply is switched off,

provided the VBAT supply is maintained.

- If LSE is selected as Auto-Wakeup unit (AWU) clock: The AWU state is not guaranteed if the VDD supply is powered off.
- If the HSE clock divided by 128 is used as the RTC clock: If the VDD power is turned off or the internal regulator is turned off (the 1.2V domain is turned off), you cannot wake up automatically via the RTC

7.3.9 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator temporization, the clock is provided to the IWDG.

7.3.10 Clock output function(MCO)

To use the MCO clock output function, the MCO pins must be configured with the appropriate multiplexing function through the GPIO and AFIO modules.

You can program the MCO output clock source to select bit RCC_CFGR.MCO[2:0] and
select one of the following four clock sources as the output clock

- SYSCLK
- HSI
- HSE
- PLL clock divided by 2

7.4 RCC registers

Tab 7.4-1 RCC register mapping

Address offset	Register	Reset value	Description
RCC base address: RCC_BA = 0x4002_1000			
0x00	RCC_CR	0x0020_XX83	Clock control register
0x04	RCC_CFGR	0x0000_0000	Clock configuration register
0x08	RCC_CIR	0x0000_0000	Clock interrupt register
0x0C	RCC_APB2RSTR	0x0000_0000	APB2 peripheral reset register
0x10	RCC_APB1RSTR	0x0000_0000	APB1 peripheral reset register
0x14	RCC_AHBENR	0x0000_0014	AHB peripheral clock enable register
0x18	RCC_APB2ENR	0x0000_0000	APB2 peripheral clock enable register
0x1C	RCC_APB1ENR	0x0000_0000	APB1 peripheral clock enable register
0x20	RCC_BDCR	0x0000_0018	Backup domain control register
0x24	RCC_CSR	0x0C00_0000	Control/status register

7.4.1 Clock control register(RCC_CR)

Register	Address offset	Access	Reset value	Description
RCC_CR	0x00	RW	0x0020_XX83	Clock control register

31	30	29	28	27	26	25	24
Reserved						PLLRDY	PLLON
23	22	21	20	19	18	17	16
Reserved	HSE_DL[2:0]			CSSON	HSEBYP	HSERDY	HSEON
15	14	13	12	11	10	9	8
HSICAL[8:1]							
7	6	5	4	3	2	1	0
HSICAL[0]	HSITRIM[3:0]				Reserved	HSIRDY	HSION

Clock control register(RCC_CR)bit description

Bit	Access	Description
[31:26]	R	Reserved
[25]	R	**PLLRDY:** PLL clock ready flag 0: PLL unlocked 1: PLL locked
[24]	RW	**PLLON:** PLL enable 0: PLL OFF 1: PLL ON Software write 1, when the system clock is not PLL, software write 0; **Note:** *When the PLL source is HSE, the hardware will automatically reset after HSE failure. Hardware automatically reset when entering stop or standby mode*
[23]	R	Reserved
[22:20]	RW	**HSE_DL[2:0]:** HSE gain control
[19]	R	Reserved
[18]	RW	**HSEBYP:** HSE passthrough enabled 0: HSE normal oscillating 1: Feed the clock directly only through IN When HSEON is 0, the software can write 1 or 0
[17]	R	**HSERDY:** HSE stability is enabled 0: HSE instability 1: HSE stabilized;
[16]	RW	**HSEON:** HSE startup enable 0: Turn off the HSE clock 1: Start HSE clock Software can write 1,Software can be written when the system clock is not HSE **Note:** *The hardware automatically cleans up after HSE failure; Hardware automatically reset when entering stop or standby mode*
[15:7]	R	**HSICAL[8:0]:** HSI calibration value; Automatically loaded from the flash memory area at startup
[6:3]	R	**HSITRIM[3:0]:** HSI trimming value; Automatically loaded from the flash memory area at startup
[2]	R	Reserved

[1]	R	**HSIRDY:** HSI stable enabled; 0: HSI not stable ; 1: HSI stable;
[0]	RW	**HSION:** HSI startup enabled; 0: turn off HSI; 1: Start HSI; Software can write 1. If the system clock is not HSI, the software can write 0. **Note:** *The hardware automatically pulls up after the failure of HSE; The hardware automatically pulls up when exiting the Stop or standby mode;*

7.4.2 Clock configuration register(RCC_CFGR)

Register	Address offset	Access	Reset value	Description
RCC_CFGR	0x04	RW	0x0000_0000	Clock configuration register

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

Clock configuration register(RCC_CFGR)bit description

Bit	Access	Description
[31:27]	R	Reserved
[26:24]	RW	**MCO[2:0]:** Microcontroller clock output 0xx: No clock 100: System clock (SYSCLK) selected 101: HSI clock selected 110: HSE clock selected 111: PLL clock divided by 2 selected
[23]	R	Reserved
[22]	RW	**USBPRE:** USB clock prescaler 0: PLL clock is divided by 2 1: PLL clock is not divided **Note:** *Write 0 or 1 before USB clock is enabled in RCC_APB1ENR register;*

[21:18]	RW	**PLLMUL[3:0]:** PLL multiplication factor You can write 1 or 0 when PLLON is 0; Note(1)(2) 0000: PLL input clock x 2 0001: PLL input clock x 3 0010: PLL input clock x 4 0011: PLL input clock x 5 0100: PLL input clock x 6 0101: PLL input clock x 7 0110: PLL input clock x 8 0111: PLL input clock x 9 1000: PLL input clock x 10 1001: PLL input clock x 11 1010: PLL input clock x 12 1011: PLL input clock x 13 1100: PLL input clock x 14 1101: PLL input clock x 15 1110: PLL input clock x 16 1111: PLL input clock x 16 **Note:** *The PLL output frequency must not exceed 96 MHz*
[17]	RW	**PLLXTPRE:** PLL clock prescaler You can write 1 or 0 when PLLON is 0; 0: Direct use of HSE 1: Use the HSE binary frequency results as the source
[16]	RW	**PLLSRC:** PLL entry clock source You can write 1 or 0 when PLLON is 0; 0: Use the HSI binary frequency results as the source 1: Use the HSE or HSI binary frequency results as the source
[15:14]	RW	**ADCPRE[1:0]:** ADC clock prescaler ADC_CLK frequency configuration, ADC_CLK from the PCLK2 frequency division 00: PCLK2 divided by 2 acts as the ADC clock 01: PCLK2 divided by 4 acts as the ADC clock 10: PCLK2 divided by 6 acts as the ADC clock 11: PCLK2 divided by 8 acts as the ADC clock
[13:11]	RW	**PPRE2[2:0]:** PCLK2 clock prescaler PCLK2 frequency configuration, PCLK2 clock from HCLK frequency division 0xx: HCLK not divided as PCLK2 100: HCLK divided by 2 as PCLK2 101: HCLK divided by 4 as PCLK2 110: HCLK divided by 8 as PCLK2 111: HCLK divided by 16 as PCLK2
[10:8]	RW	**PPRE1[2:0]:** PCLK1 clock prescaler PCLK1 frequency configuration, PCLK1 clock from HCLK frequency division 0xx: HCLK not divided as PCLK1 100: HCLK divided by 2 as PCLK1 101: HCLK divided by 4 as PCLK1 110: HCLK divided by 8 as PCLK1 111: HCLK divided by 16 as PCLK1

[7:4]	RW	**HPRE[3:0]:** HCLK clock prescaler HCLK frequency configuration, HCLK clock from SYSCLK frequency division 0xxx: SYSCLK not divided as HCLK 1000: SYSCLK divided by 2 as HCLK 1001: SYSCLK divided by 4 as HCLK 1010: SYSCLK divided by 8 as HCLK 1011: SYSCLK divided by 16 as HCLK 1100: SYSCLK divided by 64 as HCLK 1101: SYSCLK divided by 128 as HCLK 1110: SYSCLK divided by 256 as HCLK 1111: SYSCLK divided by 512 as HCLK
[3:2]	R	**SWS[1:0]:** System clock switch status The actual system clock selects the source and is updated by the hardware; If the SWS is not updated after the SW is configured, the new clock is not locked or ready 00: HSI as the system clock 01: HSE as the system clock 10: PLL as the system clock 11: unavailable
[1:0]	RW	**SW[1:0]:** System clock switch These bits are written by software, Used to indicate the selection of the System clock (SYSCLK) input source. The SWS[1:0] bit is read to determine whether the system clock source is successfully switched. 00: HSI as the system clock 01: HSE as the system clock 10: PLL as the system clock 11: unavailable **Note:** *If the HSE, which directly or indirectly serves as the system clock, fails while returning from stop or standby mode, the hardware automatically clears to 0*

Note:

1. *PLL output frequency shall not exceed 96MHz*
2. *The PLLMUL parameter of register RCC_CFGR is the multiplication factor of PLL, and PLLMUL also determines the PLL_KEY parameter inside the PLL. Not all PLLMUL can be used for different input frequencies. The PLL input clock multiplied by PLL_KEY is a valid PLLMUL configuration in the [64,320] range*

Tab 7.4-4 PLLMUL and PLL_KEY

PLLMUL	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PLL_KEY	16	24	16	20	24	14	16	18	20	22	24	13	14	15	16	16

For example, if the input frequency is 8MHZ, the frequency multiplier you choose is 12 :
8*14 = 112, $64 \leqslant 112 \leqslant 320$, Therefore, the clock configuration is stable and effective
Another example: input frequency is 24MHZ, at this time you want to set the main frequency is 96MHZ, then need to select 96/24 = 4
We know this from the table PLL_KEY = 24, :
24*24 = 576 > 320, Therefore, the configuration is incorrect.
Suggestion: When the input clock is 24MHZ, perform 2 frequency division (bit17 bits of RCC_CFGR) for 24MHZ first,

and then select the frequency doubling factor

7.4.3 Clock interrupt register(RCC_CIR)

Register	Address offset	Access	Reset value	Description
RCC_CIR	0x08	RW	0x0000_0000	Clock interrupt register

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

Clock interrupt register(RCC_CIR)bit description

Bit	Access	Description
[31:24]	R	Reserved
[23]	W	**CSSC:** Clock security system interrupt clear This bit is set by software to clear the CSSF flag. 0: No effect 1: Clear CSSF flag
[22:21]	R	Reserved
[20]	W	**PLLRDYC:** PLL ready interrupt clear This bit is set by software to clear the PLLRDYF flag 0: No effect 1: PLLRDYF cleared
[19]	W	**HSERDYC:** HSE ready interrupt clear This bit is set by software to clear the HSERDYF flag. 0: No effect 1: HSERDYF cleared
[18]	W	**HSIRDYC:** HSI ready interrupt clear This bit is set by software to clear the HSIRDYF flag. 0: No effect 1: HSIRDYF cleared
[17]	W	**LSERDYC:** LSE ready interrupt clear This bit is set by software to clear the LSERDYF flag. 0: No effect 1: LSERDYF cleared
[16]	W	**LSIRDYC:** LSI ready interrupt clear This bit is set by software to clear the LSIRDYF flag. 0: No effect 1: LSIRDYF cleared
[15:13]	R	Reserved

[12]	RW	**PLLRDYIE:** PLL ready interrupt enable Set and cleared by software to enable/disable interrupt caused by PLL lock 0: PLL lock interrupt disabled 1: PLL lock interrupt enabled
[11]	RW	**HSERDYIE:** HSE ready interrupt enable Set and cleared by software to enable/disable interrupt caused by HSE lock 0: HSE ready interrupt disabled 1: HSE ready interrupt enabled
[10]	RW	**HSIRDYIE:** HSI ready interrupt enable Set and cleared by software to enable/disable interrupt caused by HSI lock 0: HSI ready interrupt disabled 1: HSI ready interrupt enabled
[9]	RW	**LSERDYIE:** LSE ready interrupt enable Set and cleared by software to enable/disable interrupt caused by LSE lock 0: LSE ready interrupt disabled 1: LSE ready interrupt enabled
[8]	RW	**LSIRDYIE:** LSI ready interrupt enable Set and cleared by software to enable/disable interrupt caused by LSI lock 0: LSI ready interrupt disabled 1: LSI ready interrupt enabled
[7]	R	**CSSF:** Clock security system interrupt flag Set by hardware when a failure is detected in HSE Cleared by software setting the CSSC bit 0: No clock failure of HSE 1: clock failure of HSE
[6:5]	R	Reserved
[4]	R	**PLLRDYF:** PLL ready interrupt flag Set by hardware when the PLL locks and PLLRDYDIE is set. Cleared by software setting the PLLRDYC bit. 0: No clock ready interrupt caused by PLL lock 1: Clock ready interrupt caused by PLL lock
[3]	R	**HSERDYF:** HSE ready interrupt flag Set by hardware when HSE becomes stable and HSERDYDIE is set. Cleared by software setting the HSERDYC bit. 0: No clock ready interrupt of HSE 1: clock ready interrupt of HSE
[2]	R	**HSIRDYF:** HSI ready interrupt flag Set by hardware when HSI becomes stable and HSIRDYDIE is set. Cleared by software setting the HSIRDYC bit. 0: No clock ready interrupt of HSI 1: clock ready interrupt of HSI
[1]	R	**LSERDYF:** LSE ready interrupt flag Set by hardware when LSE becomes stable and LSERDYDIE is set Cleared by software setting the LSERDYC bit. 0: No clock ready interrupt of LSE 1: clock ready interrupt of LSE

[0]	R	**LSIRDYF:** LSI ready interrupt flag Set by hardware when LSI becomes stable and LSIRDYDIE is set. Cleared by software setting the LSIRDYC bit. 0: No clock ready interrupt of LSI 1: clock ready interrupt of LSI

7.4.4 APB2 peripheral reset register(RCC_APB2RSTR)

Register	Address offset	Access	Reset value	Description
RCC_APB2RSTR	0x0C	RW	0x0000_0000	APB2 peripheral reset register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADC3RST	USART1RST	TIM8RST	SPI1RST	TIM1RST	ADC2RST	ADC1RST	Reserved
7	6	5	4	3	2	1	0
Reserved	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Reserved	AFIORST

APB2 peripheral reset register(RCC_APB2RSTR)bit description

The register bit is set and cleared by the software, and the reset operation of the corresponding peripheral module on the APB2 bus is performed

0: No effect（End the reset of the corresponding peripheral module）

1: enabled the reset of the corresponding peripheral module

Bit	Access	Description
[31:16]	R	Reserved
[15]	RW	**ADC3RST:** ADC3 reset
[14]	RW	**USART1RST:** USART1 reset
[13]	RW	**TIM8RST:** TIM8 reset
[12]	RW	**SPI1RST:** SPI1 reset
[11]	RW	**TIM1RST:** TIM1 reset
[10]	RW	**ADC2RST:** ADC2 reset
[9]	RW	**ADC1RST:** ADC1 reset
[8:7]	R	Reserved
[6]	RW	**IOPERST:** GPIOE reset
[5]	RW	**IOPDRST:** GPIOD reset
[4]	RW	**IOPCRST:** GPIOC reset
[3]	RW	**IOPBRST:** GPIOB reset
[2]	RW	**IOPARST:** GPIOA reset
[1]	R	Reserved
[0]	RW	**AFIORST:** AFIO reset

7.4.5 APB1 peripheral reset register (RCC_APB1RSTR)

Register	Address offset	Access	Reset value	Description
RCC_APB1RSTR	0x10	RW	0x0000_0000	APB1 peripheral reset register

31	30	29	28	27	26	25	24
Reserved		DACRST	PWURST	BKPRST	Reserved	CANRST	Reserved
23	22	21	20	19	18	17	16
USBRST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Reserved
15	14	13	12	11	10	9	8
SPI3RST	SPI2RST	Reserved		WWDGRST	Reserved		
7	6	5	4	3	2	1	0
Reserved		TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST

APB1 peripheral reset register (RCC_APB1RSTR)bit description

The register bit is set and cleared by the software, and the reset operation of the corresponding peripheral module on the APB1 bus is performed

0: No effect（End the reset of the corresponding peripheral module）

1: enabled the reset of the corresponding peripheral module

Bit	Access	Description
[31:30]	R	Reserved
[29]	RW	**DACRST:** DAC reset
[28]	RW	**PWURST:** PWR reset
[27]	RW	**BKPRST:** BKP reset
[26]	R	Reserved
[25]	RW	**CANRST:** CAN reset
[24]	R	Reserved
[23]	RW	**USBRST:** USB reset
[22]	RW	**I2C2RST:** I2C2 reset
[21]	RW	**I2C1RST:** I2C1 reset
[20]	RW	**UART5RST:** UART5 reset
[19]	RW	**UART4ST:** UART4 reset
[18]	RW	**UART3RST:** UART3 reset
[17]	RW	**UART2ST:** UART2 reset
[16]	R	Reserved
[15]	RW	**SPI3RST:** SPI3 reset
[14]	RW	**SPI2RST:** SPI2 reset
[13:12]	R	Reserved
[11]	RW	**WWDGRST:** WWDG reset
[10:6]	R	Reserved
[5]	RW	**TIM7RST:** TIM7 reset
[4]	RW	**TIM6RST:** TIM6 reset
[3]	RW	**TIM5RST:** TIM5 reset
[2]	RW	**TIM4RST:** TIM4 reset
[1]	RW	**TIM3RST:** TIM3 reset
[0]	RW	**TIM2RST:** TIM2 reset

7.4.6 AHB peripheral clock enable register (RCC_AHBENR)

Register	Address offset	Access	Reset value	Description
RCC_AHBENR	0x14	RW	0x0000_0014	AHB peripheral clock enable register

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

AHB peripheral clock enable register (RCC_AHBENR)bit description

The register bit is set and cleared by the software, and the reset operation of the corresponding peripheral module on the AHB bus is performed

0:Disable the clock of the peripheral module

1: enabled the clock of the peripheral module

Bit	Access	Description
[31:11]	R	Reserved
[10]	RW	**SDIOEN:** SDIO clock enable
[9]	R	Reserved
[8]	RW	**QSPIEN:** QSPI clock enable
[7]	RW	**AESEN:** AES clock enable
[6]	RW	**CRCEN:** CRC clock enable
[5]	RW	**RNGEN:** RNG clock enable
[4]	RW	**FLITFEN:** FLITF clock enable Set and cleared by software to disable/enable FLITF clock during Sleep mode
[3]	R	Reserved
[2]	RW	**SRAMEN:** SRAM clock enable Set and cleared by software to disable/enable SRAM interface clock during Sleep mode
[1]	RW	**DMA2EN:** DMA2 clock enable
[0]	RW	**DMA1EN:** DMA1 clock enable

7.4.7 APB2 peripheral clock enable register(RCC_APB2ENR)

Register	Address offset	Access	Reset value	Description
RCC_APB2ENR	0x18	RW	0x0000_0000	APB2 peripheral clock enable register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADC3EN	USART1EN	TIM8EN	SPI1EN	TIM1EN	ADC2EN	ADC1EN	Reserved
7	6	5	4	3	2	1	0
Reserved	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Reserved	AFIOEN

APB2 peripheral clock enable register(RCC_APB2ENR)bit description

The register bit is set and cleared by the software, and the reset operation of the corresponding peripheral module on the APB2 bus is performed
0:Disable the clock of the peripheral module
1: enabled the clock of the peripheral module

Bit	Access	Description
[31:16]	R	Reserved
[15]	RW	**ADC3EN:** ADC3 clock enable
[14]	RW	**USART1EN:** USART1 clock enable
[13]	RW	**TIM8:** TIM8 clock enable
[12]	RW	**SPI1EN:** SPI1 clock enable
[11]	RW	**TIM1EN:** TIM1 clock enable
[10]	RW	**ADC2EN:** ADC2 clock enable
[9]	RW	**ADC1EN:** ADC1 clock enable
[8:7]	R	Reserved
[6]	RW	**IOPEEN:** GPIOE clock enable
[5]	RW	**IOPDEN:** GPIOD clock enable
[4]	RW	**IOPCEN:** GPIOC clock enable
[3]	RW	**IOPBEN:** GPIOB clock enable
[2]	RW	**IOPAEN:** GPIOA clock enable
[1]	R	Reserved
[0]	RW	**AFIOEN:** AFIO clock enable

7.4.8 RCC APB1 peripheral enabled Register(RCC_APB1ENR)

Register	Address offset	Access	Reset value	Description
RCC_APB1ENR	0x1C	RW	0x0000_0000	RCC APB1 peripheral enabled Register

31	30	29	28	27	26	25	24
Reserved		DACEN	PWREN	BKPEN	Reserved	CANEN	Reserved
23	22	21	20	19	18	17	16
USBEN	I2C2EN	I2C1EN	UART5EN	UART4EN	UART3EN	UART2EN	Reserved
15	14	13	12	11	10	9	8
SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		
7	6	5	4	3	2	1	0
Reserved		TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN

RCC APB1 peripheral enabled Register(RCC_APB1ENR)bit description

The register bit is set and cleared by the software, and the reset operation of the corresponding peripheral module on the APB1 bus is performed
0:Disable the clock of the peripheral module
1: enabled the clock of the peripheral module

Bit	Access	Description
[31:30]	R	Reserved
[29]	RW	**DACEN:** DAC clock enable
[28]	RW	**PWREN:** PWR clock enable
[27]	RW	**BKPEN:** BKP clock enable
[26]	R	Reserved

[25]	RW	**CANEN:** CAN clock enable
[24]	R	Reserved
[23]	RW	**USBEN:** USB clock enable
[22]	RW	**I2C2EN:** I2C2 clock enable
[21]	RW	**I2C1EN:** I2C1 clock enable
[20]	RW	**UART5EN:** UART5 clock enable
[19]	RW	**UART4EN:** UART5 clock enable
[18]	RW	**UART3EN:** UART5 clock enable
[17]	RW	**UART2EN:** UART5 clock enable
[16]	R	Reserved
[15]	RW	**SPI3EN:** SPI3 clock enable
[14]	RW	**SPI2EN:** SPI2 clock enable
[13:12]	R	Reserved
[11]	RW	**WWDGEN:** TIM1 clock enable
[10:6]	R	Reserved
[5]	RW	**TIM7EN:** TIM7 clock enable
[4]	RW	**TIM6EN:** TIM6 clock enable
[3]	RW	**TIM5EN:** TIM5 clock enable
[2]	RW	**TIM4EN:** TIM4 clock enable
[1]	RW	**TIM3EN:** TIM3 clock enable
[0]	RW	**TIM2EN:** TIM2 clock enable

7.4.9 Backup domain control register(RCC_BDCR)

Register	Address offset	Access	Reset value	Description
RCC_BDCR	0x20	RW	0x0000_0018	Backup domain control register

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

Backup domain control register(RCC_BDCR)bit description

Bit	Access	Description
[31:17]	R	Reserved
[16]	RW	**BDRST:** Backup domain software reset Set and cleared by software 0: Reset inactive 1: Reset the entire backup domain
[15]	RW	**RTCEN:** RTC clock enable Set and cleared by software 0: RTC clock is disabled 1: RTC clock is enabled

[14:10]	R	Reserved
[9:8]	RW	**RTCSEL[1:0]:** RTC clock source selection Set by software to select the clock source for the RTC 00: No clock 01: LSE oscillator clock used as RTC clock 10: LSI oscillator clock used as RTC clock 11: HSE oscillator clock divided by 128 used as RTC clock **Note:** *Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. The BDRST bit can be used to reset them.*
[7]	R	Reserved
[6:3]	RW	**LSE_DL[3:0]:** Gain control of the LSE oscillator
[2]	RW	**LSEBYP:** External low-speed oscillator bypass Set and cleared by software in debug mode to bypass the oscillator. This bit can be written only when the external 32 kHz oscillator is disabled 0: LSE oscillator not bypassed 1: LSE oscillator bypassed
[1]	R	**LSERDY:** External low-speed oscillator ready Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles. 0: External 32 kHz oscillator not ready 1: External 32 kHz oscillator ready
[0]	RW	**LSEON:** External low-speed oscillator enable 0: External 32 kHz oscillator OFF 1: External 32 kHz oscillator ON

7.4.10 Control/status register(RCC_CSR)

Register	Address offset	Access	Reset value	Description
RCC_CSR	0x24	RW	0x0C00_0000	Control/status register

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

Control/status register(RCC_CSR)bit description

Bit	Access	Description

[31]	R	**LPWRRSTF:** Low-power reset flag Set by hardware when a Low-power management reset occurs. Cleared by writing to the RMVF bit. 0: No Low-power management reset occurred 1: Low-power management reset occurred
[30]	R	**WWDGRSTF:** Window watchdog reset flag Set by hardware when a window watchdog reset occurs.Cleared by writing to the RMVF bit. 0: No window watchdog reset occurred 1: Window watchdog reset occurred
[29]	R	**IWDGRSTF:** Independent watchdog reset flag Set by hardware when an independent watchdog reset from V DD domain occurs.Cleared by writing to the RMVF bit. 0: No independent watchdog reset occurred 1: independent Watchdog reset occurred
[28]	R	**SFTRSTF:** Software reset flag Set by hardware when a software reset occurs,Cleared by writing to the RMVF bit 0: No software reset occurred 1: Software reset occurred
[27]	R	**PWRRSTF:** POR/PDR reset flag Set by hardware when a POR/PDR reset occurs. Cleared by writing to the RMVF bit. 0: No POR/PDR reset occurred 1: POR/PDR reset occurred
[26]	R	**PINRSTF:** PIN reset flag Set by hardware when a reset from the NRST pin occurs.Cleared by writing to the RMVF bit. 0: No reset from NRST pin occurred 1: Reset from NRST pin occurred
[25]	R	Reserved
[24]	RW	**RMVF:** Remove reset flag Set by software to clear the reset flags 0: No effect 1: Clear the reset flags
[23:2]	R	Reserved
[1]	R	**LSIRDY:** Internal low-speed oscillator ready Hardware setting and zeroing indicate when the LSI is stable. 0: LSI clock is unstable 1: LSI clock is stable
[0]	RW	**LSION:** Internal low-speed oscillator enable Set and clear by software. 0: Disable the LSI clock 1: start LSI clock

8 General-purpose I/O (GPIO)

8.1 GPIO functional description

Each of the general-purpose I/O ports has two 32-bit configuration registers(GPIOx_CRL, GPIOx_CRH), wo 32-bit data registers(GPIOx_IDR, GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 16-bit reset register (GPIOx_BRR)and a 32-bit locking register(GPIOx_LCKR)。

Each port bit of the General Purpose IO (GPIO) Ports, can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words (half-word or byte accesses are not allowed). The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIO registers. This way, there is no risk that an IRQ occurs between the read and the modify access.

Fig 8.1-1 Basic structure of a standard I/O port bit

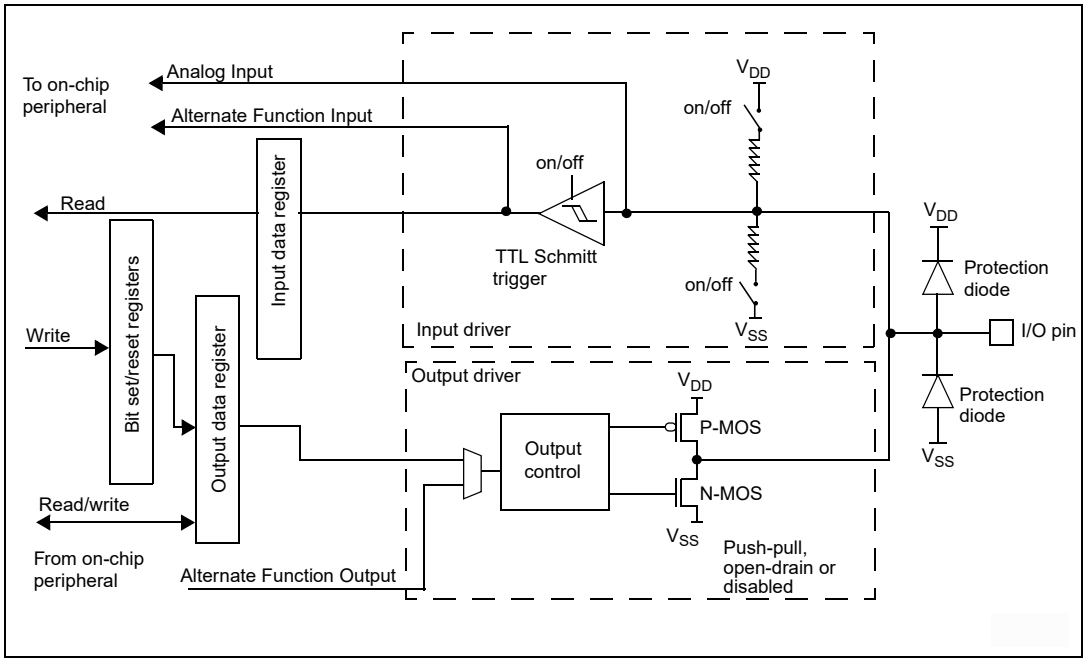
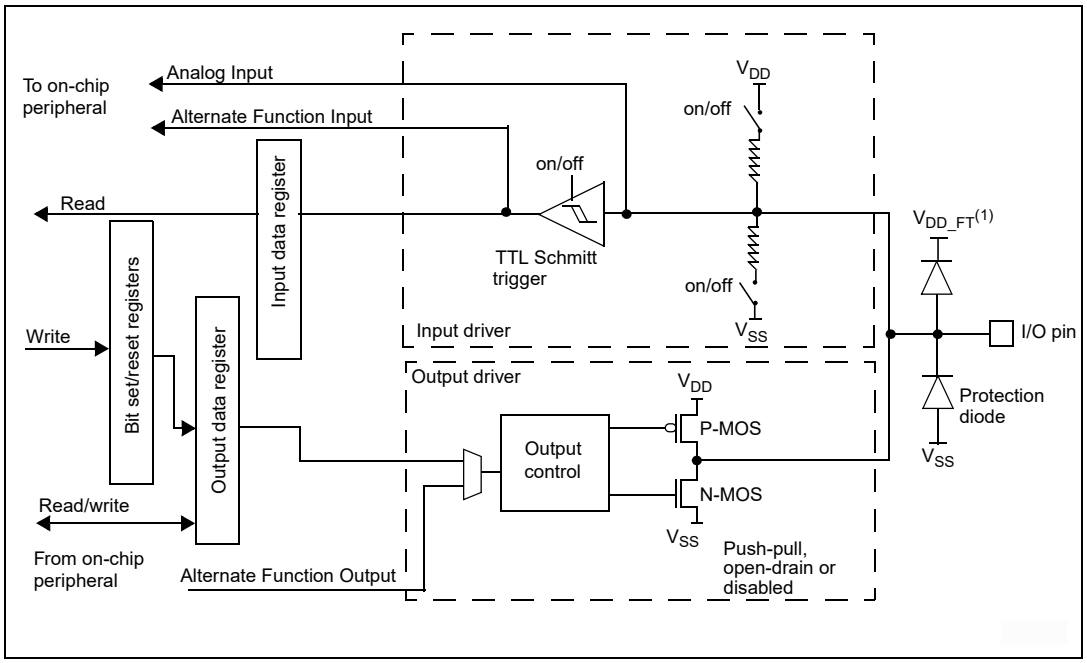


Fig 8.1-2 Basic structure of a 5-Volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to 5-Volt tolerant I/Os, and different from V_{DD}.

Tab 8.1-1 Port bit configuration table

Configuration mode		CNF1	CNF0	MODE1/ MODE0	PxODR register
General purpose output	Push-pull	0	0	01	0 or 1
	Open-drain		1	10	0 or 1
Alternate Function output	Push-pull	1	0	11	disable
	Open-drain		1	See table below	disable
Input	Analog	0	0	00	disable
	Input floating		1		disable
	Input pull-down	1	0		0
	Input pull-up				1

Tab 8.1-2 Output MODE bits

MODE【1:0】	Meaning
00	Input mode (reset state)
01	Maximum output speed 10 MHz
10	Maximum output speed 2 MHz
11	Maximum output speed 50 MHz

8.1.1 General-purpose I/O

During and just after reset, the alternate functions are not active and the I/O ports are configured in Input Floating mode (CNFx[1:0]=01b, MODEx[1:0]=00b).

When configured as output, the value written to the Output Data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in Push-Pull mode or Open-Drain mode (only the N-MOS is activated when outputting 0).

The Input Data register (GPIOx_IDR) captures the data present on the I/O pin at every APB2 clock cycle.

All GPIO pins have an internal weak pull-up and weak pull-down that can be activated or not when configured as input.

8.1.2 Atomic bit set or reset

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify only one or several bits in a single atomic APB2 write access. This is achieved by programming to ‘1’ the Bit Set/Reset Register (GPIOx_BSRR,or for reset only GPIOx_BRR) to select the bits to modify. The unselected bits will not be modified.

8.1.3 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode. For more information on external interrupts, refer to Section：External interrupt/event controller and Section： Wakeup event management

8.1.4 Alternate functions (AF)

It is necessary to program the Port Bit Configuration Register before using a default alternate function.

- For alternate function inputs, the port must be configured in Input mode (floating, pull- up or pull-down) and the input pin must be driven externally.

Note: *It is also possible to emulate the AFI input pin by software by programming the GPIO controller. In this case, the port should be configured in Alternate Function Output mode. And obviously, the corresponding port should not be driven externally as it will be driven by the software using the GPIO controller.*

- For alternate function outputs, the port must be configured in Alternate Function Output mode (Push-Pull or Open-Drain).
- For bidirectional Alternate Functions, the port bit must be configured in Alternate Function Output mode (Push-Pull or Open-Drain). In this case the input driver is configured in input floating mode

If a port bit is configured as Alternate Function Output, this disconnects the output register and connects the pin to the output signal of an on-chip peripheral.

If software configures a GPIO pin as Alternate Function Output, but peripheral is not activated, its output is not specified.

8.1.5 Software remapping of I/O alternate functions

To optimize the number of peripheral I/O functions for different device packages, it is possible to remap some alternate functions to some other pins. This is achieved by software, by programming the corresponding registers . In that case, the alternate functions are no longer mapped to their original assignations.

8.1.6 GPIO locking mechanism

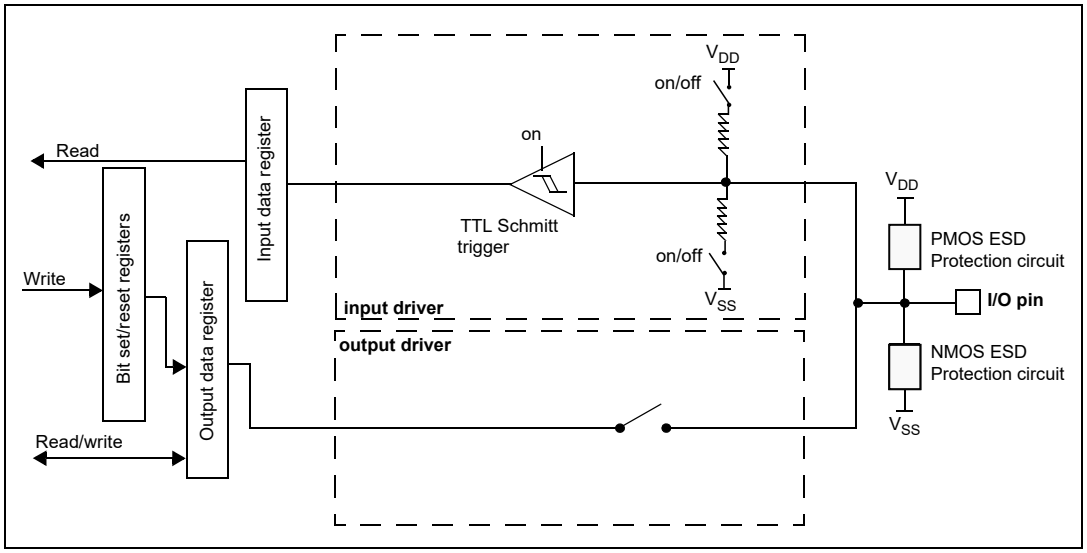
The locking mechanism allows the IO configuration to be frozen. When the LOCK sequence has been applied on a port bit, it is no longer possible to modify the value of the port bit until the next reset

8.1.7 Input configuration

When the I/O Port is programmed as Input:

- The Output Buffer is disabled
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are activated or not depending on input configuration (pull-up, pull-down or floating)
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register obtains the I/O State.

Fig 8.1-3 Input floating/pull up/pull down configurations



Note: *VDD_FT is a potential specific to 5-Volt tolerant I/Os, and different from VDD*

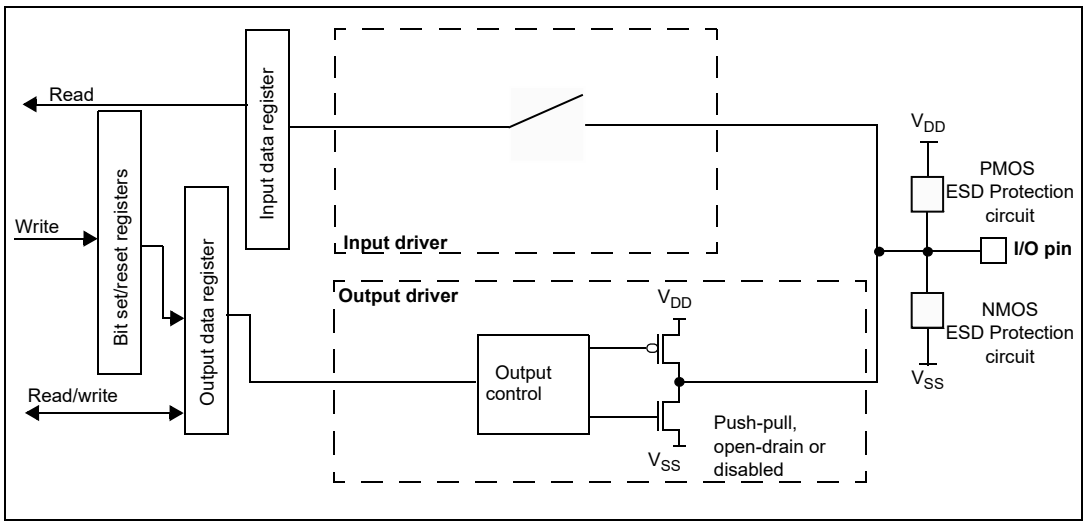
8.1.8 Output configuration

When the I/O Port is programmed as Output:

- The Output Buffer is enabled:
  - Open Drain Mode: A “0” in the Output register activates the NMOS while a “1” in the Output register leaves the port in high impedance(the PMOS is never activated)
  - Push-Pull Mode: A “0” in the Output register activates the NMOS while a “1” in the Output register activates the PMOS
- The Schmitt Trigger Input is disabled.
- The weak pull-up and pull-down resistors are disabled.
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register gets the I/O state in open drain mode

- A read access to the Output Data register gets the last written value in Push-Pull mode

Fig 8.1-4 Output configuration



Note: *VDD_FT is a potential specific to 5-Volt tolerant I/Os, and different from VDD*

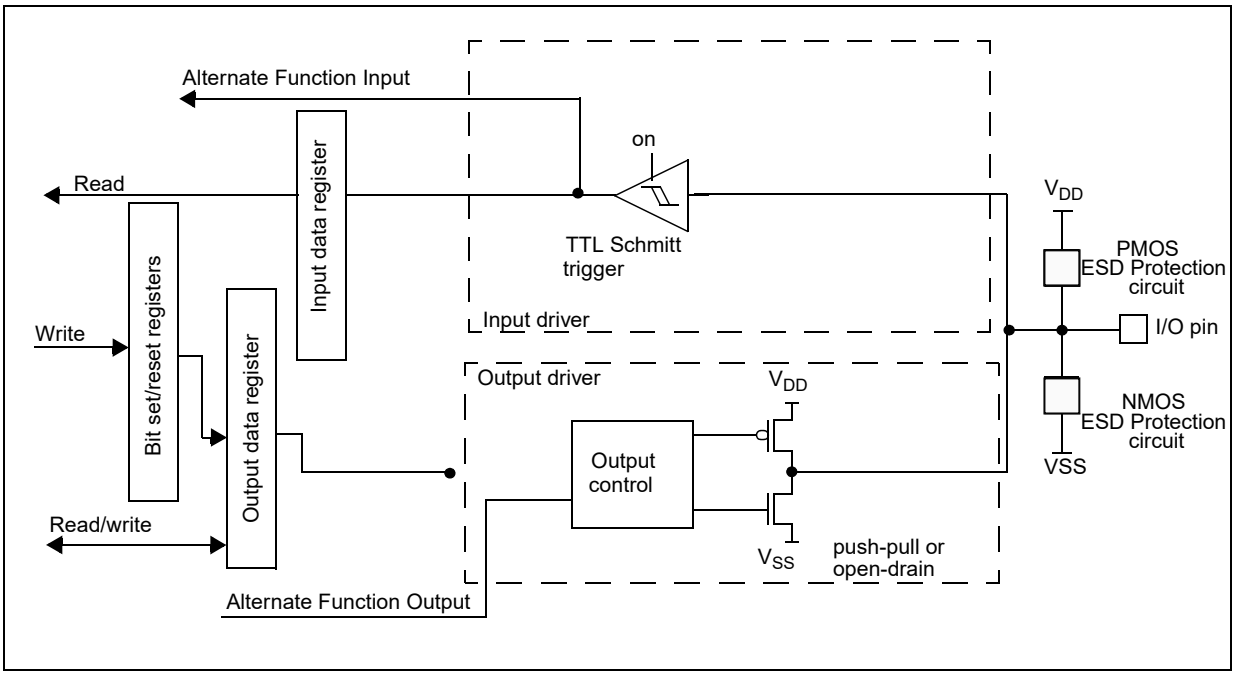
8.1.9 Alternate function configuration

When the I/O Port is programmed as Alternate Function:

- The Output Buffer is turned on in Open Drain or Push-Pull configuration
- The Output Buffer is driven by the signal coming from the peripheral (alternate function out)
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are disabled.
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register gets the I/O state in open drain mode
- A read access to the Output Data register gets the last written value in Push-Pull mode

A set of Alternate Function I/O registers allows the user to remap some alternate functions to different pins.

Fig 8.1-5 Alternate function configuration



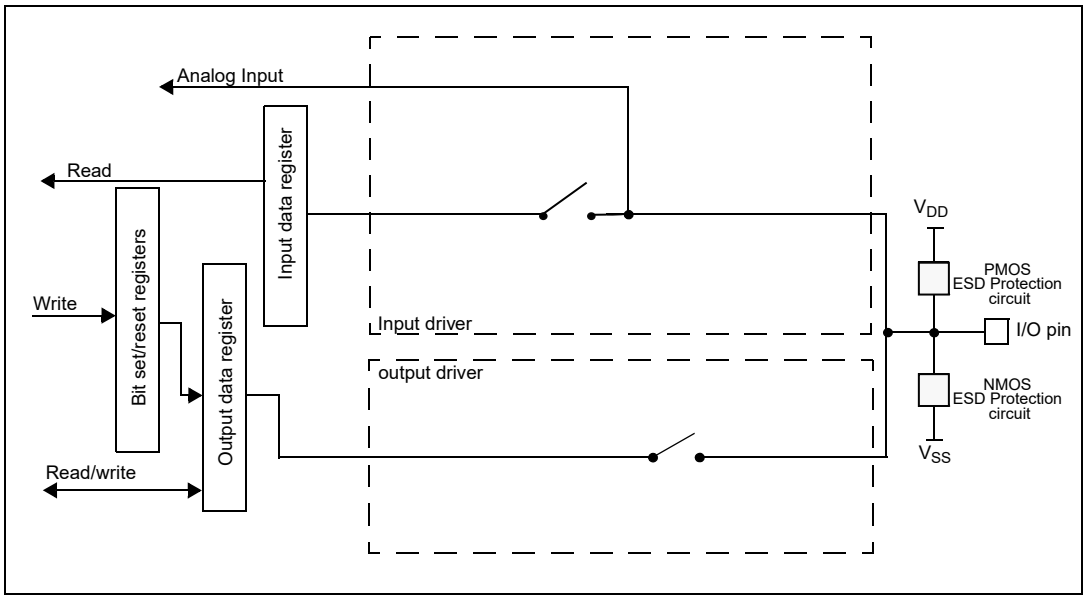
Note: *VDD_FT is a potential specific to 5-Volt tolerant I/Os, and different from VDD*

8.1.10 Analog configuration

When the I/O Port is programmed as Analog configuration:

- The Output Buffer is disabled.
- The Schmitt Trigger Input is de-activated providing zero consumption for every analog value of the I/O pin. The output of the Schmitt Trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled
- Read access to the Input Data Register gets the value “0”.

Fig 8.1-6 The high impedance-analog configuration of the I/O Port bit



Note: *VDD_FT is a potential specific to 5-Volt tolerant I/Os, and different from VDD*

8.1.11 GPIO configurations for device peripherals

Tab 8.1-3 GPIO configurations for device peripherals

Pinout	Configuration	GPIO configuration
Advanced timers TIM1 and TIM8		
TIM1/8_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull
TIM1/8_CHxN	Complementary output channel x	Alternate function push-pull
TIM1/8_BKIN	Break input	Input floating
TIM1/8_ETR	External trigger timer input	Input floating
General-purpose timers TIM2/3/4/5		
TIM2/3/4/5_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull
TIM2/3/4/5_ETR	External trigger timer input	Input floating
USARTs		
USARTx_TX	Full duplex	Alternate function push-pull
	Half duplex synchronous mode	Alternate function push-pull
USARTx_RX	Full duplex	Input floating / Input pull-up
	Half duplex synchronous mode	Not used. Can be used as a general IO
USARTx_CK	Synchronous mode	Alternate function push-pull
USARTx_RTS	Hardware flow control	Alternate function push-pull
USARTx_CTS	Hardware flow control	Input floating / Input pull-up
SPI		
SPIx_SCK	Master	Alternate function push-pull
	Slave	Input floating
SPIx_MOSI	Full duplex/Master	Alternate function push-pull
	Full duplex/Slave	Input floating / Input pull-up

(Continuation)

Pinout	Configuration	GPIO configuration
	Simplex bidirectional data wire/Master	Alternate function push-pull
	Simplex bidirectional data wire/Slave	Not used. Can be used as a general IO
SPIx_MISO	Full duplex/Master	Input floating / Input pull-up
	Full duplex/Slave	Alternate function push-pull
	Simplex bidirectional data wire/Master	Not used. Can be used as a general IO
	Simplex bidirectional data wire/Slave	Alternate function push-pull
SPIx_NSS	Hardware master/Slave	Input floating/ Input pull-up / Input pull-down
	Hardware master/NSS output enabled	Alternate function push-pull
	Software	Not used. Can be used as a general IO
I2S		
I2Sx_WS	Master	Alternate function push-pull
	Slave	Input floating
I2Sx_CK	Master	Alternate function push-pull
	Slave	Input floating
I2Sx_SD	Transmitter	Alternate function push-pull
	Receiver	Input floating/ Input pull-up / Input pull-down
I2Sx_MCK	Master	Alternate function push-pull
	Slave	Not used. Can be used as a general IO
I2C		
I2Cx_SCL	I2C clock	Alternate function open drain
I2Cx_SDA	I2C Data	Alternate function open drain
BxCAN		
CAN_TX		Alternate function push-pull
CAN_RX		Input floating / Input pull-up
USB		
USB_DM / USB_DP		As soon as the USB is enabled, these pins are automatically connected to the USB internal transceiver.
SDIO		
SDIO_CK		Alternate function push-pull
SDIO_CMD		Alternate function push-pull
SDIO[D7:D0]		Alternate function push-pull
ADC/DAC(The GPIO configuration of the ADC inputs should be analog.)		
ADC/DAC		Analog
Other IOs		
TAMPER-RTC	RTC output	Forced by hardware when configuring the BKP_CR and BKP_RTCCR registers
	Tamper event input	
MCO	Clock output	Alternate function push-pull
EXTI input lines	External input interrupts	Input floating/ Input pull-up / Input pull-down

8.2 GPIO Register

Tab 8.2-1 GPIO Register Map

offset	Register	Reset value	Description
GPIOA: GPIOA_BA = 0x4001_0800 GPIOB: GPIOB_BA = 0x4001_0C00 GPIOC: GPIOC_BA = 0x4001_1000 GPIOD: GPIOD_BA = 0x4001_1400 GPIOE: GPIOD_BA = 0x4001_1800			
0x00	GPIOx_CRL	0x4444_4444	Port configuration register low
0x04	GPIOx_CRH	0x4444_4444	Port configuration register high
0x08	GPIOx_IDR	0x0000_0000	Port input data register
0x0C	GPIOx_ODR	0x0000_0000	Port output data register
0x10	GPIOx_BSRR	0x0000_0000	Port bit set/reset register
0x14	GPIOx_BRR	0x0000_0000	Port bit reset register
0x18	GPIOx_LCKR	0x0000_0000	Port configuration lock register

8.2.1 Port configuration register low(GPIOx_CRL, x = A, B, C, D, E)

Register	Address offset	Access	Reset value	Description
GPIOx_CRL	0x00	RW	0x4444_4444	Port configuration register low (x = A, B, C, D, E)

31	30	29	28	27	26	25	24
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]	
23	22	21	20	19	18	17	16
CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
15	14	13	12	11	10	9	8
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]	
7	6	5	4	3	2	1	0
CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	

Port configuration register low(GPIOx_CRL, x = A, B, C, D, E)bit description

Bit	Access	Description
[31:30] [27:26] [23:22] [19:18] [15:14] [11:10] [7:6] [3:2]	RW	**CNFy[1:0]:** Port x configuration bit y(y = 0 ~ 7) These bits are written by software to configure the corresponding I/O port, Refer to: Port bit configuration table In input mode (MODE[1:0]=00): 00: Analog mode 01: Floating input (reset state) 10: Input with pull-down 11: Input with pull-up In output mode(MODE[1:0]>00): 00: General purpose output push-pull 01: General purpose output Open-drain 10: Alternate function output Push-pull 11: Alternate function output Open-drain

[29:28] [25:24] [21:20] [17:16] [13:12] [9:8] [5:4] [1:0]	RW	**MODEy[1:0]:** Port x mode bit y(y = 0 ~ 7) These bits are written by software to configure the corresponding I/O port, Refer to: Port bit configuration table 00: Input mode (reset state) 01: Output mode, max speed 10 MHz. 10: Output mode, max speed 2 MHz. 11: Output mode, max speed 50 MHz

8.2.2 Port configuration register high(GPIOx_CRH, x = A, B, C, D, E)

Register	Address offset	Access	Reset value	Description
GPIOx_CRH	0x04	RW	0x4444_4444	Port configuration register high (GPIOx_CRH,x = A, B, C, D, E)

31	30	29	28	27	26	25	24
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]	
23	22	21	20	19	18	17	16
CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
15	14	13	12	11	10	9	8
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]	
7	6	5	4	3	2	1	0
CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	

Port configuration register high (GPIOx_CRL, x = A, B, C, D, E)bit description

Bit	Access	Description
[31:30] [27:26] [23:22] [19:18] [15:14] [11:10] [7:6] [3:2]	RW	**CNFy[1:0]:** Port x configuration bit y(y = 8 ~ 15) These bits are written by software to configure the corresponding I/O port, Refer to: Port bit configuration table In input mode (MODE[1:0]=00): 00: Analog mode 01: Floating input (reset state) 10: Input with pull-down 11: Input with pull-up In output mode (MODE[1:0] > 00) 00: General purpose output push-pull 01: General purpose output Open-drain 10: Alternate function output Push-pull 11: Alternate function output Open-drain

[29:28] [25:24] [21:20] [17:16] [13:12] [9:8] [5:4] [1:0]	RW	**MODEy[1:0]:** Port x mode bit y(y = 8 ~ 15) These bits are written by software to configure the corresponding I/O port，Refer to：Port bit configuration table 00: Input mode (reset state) 01: Output mode, max speed 10 MHz 10: Output mode, max speed 2 MHz. 11: Output mode, max speed 50 MHz.

8.2.3 Port input data register(GPIOx_IDR, x = A, B, C, D, E)

Register	Address offset	Access	Reset value	Description
GPIOx_IDR	0x08	R	0x0000_0000	Port input data register(GPIOx_IDR,x = A, B, C, D, E)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8
7	6	5	4	3	2	1	0
IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0

Port input data register(GPIOx_IDR, x = A, B, C, D, E)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.
[15:0]	R	**IDRy:** Port input data(y = 0 ~ 15) These bits are read only and can be accessed in Word mode（16 bit）only. They contain the input value of the corresponding I/O port.

8.2.4 Port output data register(GPIOx_ODR, x = A, B, C, D, E)

Register	Address offset	Access	Reset value	Description
GPIOx_ODR	0x0C	R	0x0000_0000	Port output data register (GPIOx_ODR, x = A, B, C, D, E)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8
7	6	5	4	3	2	1	0
ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0

Port output data register(GPIOx_ODR, x = A, B, C, D, E)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.

[15:0]	R	**ODRy:** Port output data (y = 0 ~ 15) These bits can be read and written by software and can be accessed in Word mode（16 bit）only. They contain the input value of the corresponding I/O port. **Note:** *For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. G).*

8.2.5 Port bit set/reset register(GPIOx_BSRR, x = A, B, C, D, E)

Register	Address offset	Access	Reset value	Description
GPIOx_BSRR	0x10	W	0x0000_0000	Port bit set/reset register (GPIOx_BSRR,x = A, B, C, D, E)

31	30	29	28	27	26	25	24
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8
23	22	21	20	19	18	17	16
BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
15	14	13	12	11	10	9	8
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8
7	6	5	4	3	2	1	0
BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0

Port bit set/reset register(GPIOx_BSRR, x = A, B, C, D, E)bit description

Bit	Access	Description
[31:16]	W	**BRy:** Port x Reset bit y(y = 0 ~ 15) These bits are write-only and can be accessed in Word mode(16 bit) only. 0: No action on the corresponding ODRy bit 1: Reset the corresponding ODRy bit 0 **Note:** *If both BSy and BRy are set, BSy has priority.*
[15:0]	W	**BSy:** Pin y set bit, y = 0 ~ 15 These bits are write-only and can be accessed in Word mode（16 bit）only. 0: No action on the corresponding ODRy bit 1: Set the corresponding ODRy bit 1

8.2.6 Port bit reset register(GPIOx_BRR, x = A, B, C, D, E)

Register	Address offset	Access	Reset value	Description
GPIOx_BRR	0x14	W	0x0000_0000	Port bit reset register (GPIOx_BRR,x = A, B, C, D, E)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8
7	6	5	4	3	2	1	0
BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0

Port bit reset register(GPIOx_BRR, x = A, B, C, D, E)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	W	**BRy:** Port x Reset bit y(y = 0 ~ 15) These bits are write-only and can be accessed in Word mode（16 bit）only 0: No action on the corresponding ODRy bit 1: Reset the corresponding ODRy bit 0

8.2.7 Port configuration lock register(GPIOx_LCKR, x = A, B, C, D, E)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit it is no longer possible to modify the value of the port bit until the next reset. Each lock bit freezes the corresponding 4 bits of the control register (CRL, CRH)

Register	Address offset	Access	Reset value	Description
GPIOx_LCKR	0x18	RW	0x0000_0000	Port configuration lock register (GPIOx_LCKR,x = A, B, C, D, E)

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

Port configuration lock register(GPIOx_LCKR, x = A, B, C, D)bit description

Bit	Access	Description
[31:17]	R	Reserved
[16]	RW	**LCKK:** Lock key This bit can be read anytime. It can only be modified using the Lock Key Writing Sequence 0: Port configuration lock key not active 1: Port configuration lock key active. GPIOx_LCKR register is locked until the next reset. LOCK key writing sequence: Write 1 - Write 0 - Write 1 - Read 0 - Read 1 The last read is optional but confirms that the lock is active, During the LOCK Key Writing sequence, the value of LCK[15:0] must not change, Any error in the lock sequence will abort the lock
[y]	RW	**LCKy:** Port x Lock bit y(y = 0 ~ 15) These bits are read write but can only be written when the LCKK bit is 0. 0: Port configuration not locked 1: Port configuration locked

9 Alternate function I/O (AFIO)

9.1 AFIO functional description

To optimize the number of peripherals available in 64-pin or 100-pin packages, some alternate functions can be remapped to other pins. This can be done in software by programming the AF remap and debugging the I/O configuration register (AFIO_MAPR). In this case, the AF is no longer mapped to its original allocation. When an I/O port is programmed for an alternate function:

9.1.1 Using OSC32_IN/OSC32_OUT pins as GPIO ports PC14/PC15

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

Note: *The PC14/PC15 GPIO functionality is lost when the 1.8 V domain is powered off (by entering standby mode) or when the backup domain is supplied by V BAT (VDD no more supplied). In this case the IOs are set in analog mode.*

9.1.2 Using OSC_IN/OSC_OUT pins as GPIO ports PD0/PD1

The HSE oscillator pins OSC_IN/OSC_OUT can be used as general-purpose I/O PD0/PD1 by programming the PD01_REMAP bit in the AF remap and debug I/O configuration register(AFIO_MAPR).

This remap is available only on 48- and 64-pin packages (PD0 and PD1 are available on 100-pin packages, no need for remapping).

Note: *The external interrupt/event function is not remapped. PD0 and PD1 cannot be used for external interrupt/event generation on 48- and 64-pin packages.*

9.1.3 CAN1 alternate function remapping

The CAN signals can be mapped on Port A, Port B or Port D as shown in Table 9.1-1. For port D, remapping is not possible in devices delivered in 48- and 64-pin packages.

Tab 9.1-1 CAN1 alternate function remapping

Alternate function	CAN_REMAP[1:0] = “00”	CAN_REMAP[1:0] = “10”	CAN_REMAP[1:0] = “11”
CAN_RX	PA11	PB8	PD0
CAN_TX	PA12	PB9	PD1

Note:
CAN_REMAP[1:0] = “11”,This remapping is available only on 100-pin packages, when PD0 and PD1 are not remapped on OSC-IN and OSC-OUT.

9.1.4 SWD alternate function remapping

The debug interface signals are mapped on the GPIO ports as shown in 9.1-2

Tab 9.1-2 Debug interface signals

Alternate function	GPIO port
SWDIO	PA13
SWCLK	PA14
TRACESWO	PB3

To optimize the number of free GPIOs during debugging, this mapping can be configured in different ways by programming the SWJ_CFG[1:0] bits in the AF remap and debug I/O configuration register (AFIO_MAPR). Refer to 9.1-3

Tab 9.1-3 Debug port mapping

SWJ_CFG[2:0]	Available debug ports	SWJ I/O pin assigned				
		PA13 / JTMS / SWDIO	PA14 / JTCK /SWCLK	PA15 / JTDI	PB3 / JTDO / TRACESWO	PB4 / NJTRST
010	SW-DP Enabled	X	X	Free	Free	Free
100	SW-DP Disabled	Free	Free	Free	Free	Free
Other	Forbidden	-	-	-	-	-

9.1.5 ADC alternate function remapping

Tab 9.1-4 ADC1 external trigger injected conversion alternate function remapping

Alternate function	ADC1_ETRGINJ_REMAP = 0	ADC1_ETRGINJ_REMAP = 1
ADC1 external trigger injected conversion	ADC1 external trigger injected conversion is connected to EXTI15	ADC1 external trigger injected conversion is connected to TIM8_CH4

Tab 9.1-5 ADC1 external trigger regular conversion alternate function remapping

Alternate function	ADC1_ETRGREG_REMAP= 0	ADC1_ETRGREG_REMAP = 1
ADC1 external trigger regular conversion	ADC1 external trigger regular conversion is connected to EXTI11	ADC1 external trigger regular conversion is connected to TIM8_TRGO

Tab 9.1-6 ADC2 external trigger injected conversion alternate function remapping

Alternate function	ADC2_ETRGINJ_REMAP = 0	ADC2_ETRGINJ_REMAP = 1
ADC2 external trigger injected conversion	ADC2 external trigger injected conversion is connected to EXTI15	ADC2 external trigger injected conversion is connected to TIM8_CH4

Tab 9.1-7 ADC2 external trigger regular conversion alternate function remapping

Alternate function	ADC2_ETRGREG_REG = 0	ADC2_ETRGREG_REG = 1
ADC2 external trigger regular conversion	ADC2 external trigger regular conversion is connected to EXTI11	ADC2 external trigger regular conversion is connected to TIM8_TRGO

9.1.6 Timer alternate function remapping

Timer 4 channels 1 to 4 can be remapped from Port B to Port D , Other timer remapping possibilities are listed in Table 9.1-8 to Table 46.

Tab 9.1-8 TIM5 alternate function remapping

Alternate function	TIM5CH4_IREMAP = 0	TIM5CH4_IREMAP = 1
TIM5_CH4	TIM5 Channel4 is connected to PA3	LSI internal clock is connected to TIM5_CH4 input for calibration purpose.

Tab 9.1-9 TIM4 alternate function remapping

Alternate function	TIM4_REMAP = 0	TIM4_REMAP = 1 [1]
TIM4_CH1	PB6	PD12
TIM4_CH2	PB7	PD13
TIM4_CH3	PB8	PD14
TIM4_CH4	PB9	PD15

Note: *1. Remap available only for 100-pin package.*

Tab 9.1-10 TIM3 alternate function remapping

Alternate function	TIM3_REMAP[1:0] = “00” (no remap)	TIM3_REMAP[1:0] = “10” (partial remap)	TIM3_REMAP[1:0] = “11” (full remap) [1]
TIM3_CH1	PA6	PB4	PC6
TIM3_CH2	PA7	PB5	PC7
TIM3_CH3	PB0		PC8
TIM3_CH4	PB1		PC9

Note: *1. Remap available only for 64-pin, 100-pin packages.*

Tab 9.1-11 TIM2 alternate function remapping

Alternate function	TIM2_REMAP[1:0] = “00” (no remap)	TIM2_REMAP[1:0] = “01” (partial remap)	TIM2_REMAP[1:0] = “10” (partial remap)	TIM2_REMAP[1:0] = “11” (full remap) [1]
TIM2_CH1_ETR [2]	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

Note: *TIM_CH1 and TIM_ETR share the same pin but cannot be used at the same time (which is why we have this notation: TIM2_CH1_ETR)*

Tab 9.1-12 TIM1 alternate function remapping

Alternate function	TIM1_REMAP[1:0] = “00” (no remap)	TIM1_REMAP[1:0] = “01” (partial remap)	TIM1_REMAP[1:0] = “11” (partial remap) [1]
TIM1_ETR	PA12		PE7
TIM1_CH1	PA8		PE9
TIM1_CH2	PA9		PE11
TIM1_CH3	PA10		PE13
TIM1_CH4	PA11		PE14
TIM1_BKIN	PB12 [2]	PA6	PE15
TIM1_CH1N	PB13	PA7	PE8
TIM1_CH2N	PB14 [2]	PB0	PE10
TIM1_CH3N	PB15 [2]	PB1	PE12

Note: *1. Remap available only for 100-pin*

9.1.7 USART alternate function remapping

Tab 9.1-13 USART3 remapping

Alternate function	USART3_REMAP[1:0] = “00” (no remap)	USART3_REMAP[1:0] = “01” (partial remap)[1]	USART3_REMAP[1:0] = “11” (full remap) [2]
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

Note: *1. Remap available only for 64-pin, 100-pin packages.*

2. Remap available only for 100-pin packages.

Tab 9.1-14 USART2 remapping

Alternate function	USART2_REMAP = 0	USART2_REMAP = 1[1]
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

Note: *1. Remap available only for 100-pin packages*

Tab 9.1-15 USART1 remapping

Alternate function	USART1_REMAP = 0	USART1_REMAP = 1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

9.1.8 I2C1 alternate function remapping

Tab 9.1-16 I2C1 remapping

Alternate function	I2C1_REMAP = 0	I2C1_REMAP = 1[1]
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

9.1.9 SPI1 alternate function remapping

Tab 9.1-17 SPI1 remapping

Alternate function	SPI1_REMAP = 0	SPI1_REMAP = 1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

9.2 AFIO Register

Tab 9.2-1 AFIO Register map

offset	Register	Reset value	Description
AFIO base address: AFIO_BA = 0x4001_0000			
0x00	AFIO_EVCR	0x0000_0000	Event control register
0x04	AFIO_MAPR	0x0000_0000	Alternate remap and debug I/O configuration register
0x08	AFIO_EXTICR1	0x0000_0000	External interrupt configuration register 1
0x0C	AFIO_EXTICR2	0x0000_0000	External interrupt configuration register 2
0x10	AFIO_EXTICR3	0x0000_0000	External interrupt configuration register 3
0x14	AFIO_EXTICR4	0x0000_0000	External interrupt configuration register 4

9.2.1 AFIO Event control register(AFIO_EVCR)

Register	Address offset	Access	Reset value	Description
AFIO_EVCR	0x00	RW	0x0000_0000	AFIO Event control register

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

AFIO Event control register(AFIO_EVCR)bit description

Bit	Access	Description
[31:8]	R	Reserved,must be kept at reset value.
[7]	RW	**EVOE:** Event output enable Set and cleared by software. When set the EVENTOUT Cortex output is connected to the I/O selected by the PORT[2:0] and PIN[3:0] bits
[6:4]	RW	**PORT[2:0]:** Port selection Set and cleared by software, Select the port used to output the Cortex EVENTOUT signal. **Note:** *The EVENTOUT signal output capability is not extended to ports PF and PG.* 000: PA selected 001: PB selected 010: PC selected 011: PD selected 100: PE selected
[3:0]	RW	**PIN[3:0]:** Pin selection (x = A .. E) Set and cleared by software, Select the pin used to output the Cortex EVENTOUT signal. 0000: selected Px0 0001: selected Px1 0010: selected Px2 0011: selected Px3 ... 1111: selected Px15

9.2.2 Alternate remap and debug I/O configuration register(AFIO_MAPR)

Register	Address offset	Access	Reset value	Description
AFIO_MAPR	0x00	RW	0x0000_0000	Alternate remap and debug I/O configuration 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
OP_REMAP			Reserved								ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	TIM5CH4_IREMAP	PD01_REMAP	CAN_REMAP[1:0]		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP

Alternate remap and debug I/O configuration register(AFIO_MAPR)bit description

Bit	Access	Description
[31:29]	RW	Bit 31: OP3_REMAP = 0: OP1+,OP1-,OP1O, The three signals are not remapped to GPIO OP3_REMAP = 1: OP1+,OP1-,OP1O, The three signals are remapped to GPIO Bit 30: OP2_REMAP = 0: OP2+,OP2-,OP2O, The three signals are not remapped to GPIO OP2_REMAP = 1: OP2+,OP2-,OP2O, The three signals are remapped to GPIO Bit 29: OP1_REMAP = 0: OP1+,OP1-,OP1O, The three signals are not remapped to GPIO OP1_REMAP = 1: OP1+,OP1-,OP1O, The three signals are remapped to GPIO
[28:21]	R	Reserved,Must remain clear
[20]	RW	**ADC2_ETRGREG_REMAP:** ADC2 external trigger regular conversion remapping Set and cleared by software. This bit controls the trigger input connected to ADC2 external trigger regular conversion. When this bit is reset, the ADC2 external trigger regular conversion is connected to EXTI11. When this bit is set, the ADC2 external event regular conversion is connected to TIM8_TRGO.
[19]	RW	**ADC2_ETRGINJ_REMAP:** ADC2 external trigger injected conversion remapping Set and cleared by software.This bit controls the trigger input connected to ADC2 external trigger injected conversion. 0: The ADC2 external trigger injected conversion is connected to EXTI15 1: The ADC2 external event injected conversion is connected to TIM8_Channel4
[18]	RW	**ADC1_ETRGREG_REMAP:** ADC1 external trigger regular conversion remapping Set and cleared by software. This bit controls the trigger input connected to ADC1 External trigger regular conversion. 0: The ADC1 External trigger regular conversion is connected to EXTI11. 1: The ADC1 External Event regular conversion is connected to TIM8_TRGO.
[17]	RW	**ADC1_ETRGINJ_REMAP:** ADC1 external trigger injected conversion remapping Set and cleared by software. This bit controls the trigger input connected to ADC1 0: ADC1 External trigger injected conversion is connected to EXTI15 1: ADC1 External Event injected conversion is connected to TIM8_Channel4

[16]	RW	**TIM5CH4_IREMAP:** TIM5 channel4 internal remap Set and cleared by software. This bit controls the TIM5_CH4 internal mapping 0: TIM5_CH4 is connected to PA3 1: Set the LSI internal clock is connected to TIM5_CH4 input for calibration purpose.
[15]	RW	**PD01_REMAP:** Port D0/Port D1 mapping on OSC_IN/OSC_OUT This bit is set and cleared by software. It controls the mapping of PD0 and PD1 GPIO functionality. When the HSE oscillator is not used (application running on internal 8 MHz RC) PD0 and PD1 can be mapped on OSC_IN and OSC_OUT. This is available only on 48- and 64-pin packages (PD0 and PD1 are available on 100-pin packages, no need for remapping) 0: No remapping of PD0 and PD1 1: PD0 remapped on OSC_IN, PD1 remapped on OSC_OUT
[14:13]	RW	**CAN_REMAP[1:0]:** CAN alternate function remapping These bits are set and cleared by software. They control the mapping of alternate functions CAN_RX and CAN_TX in devices with a single CAN interface 00: CAN_RX mapped to PA11, CAN_TX mapped to PA12 01: Not used 10: CAN_RX mapped to PB8, CAN_TX mapped to PB9 11: CAN_RX mapped to PD0, CAN_TX mapped to PD1
[12]	RW	**TIM4_REMAP:** TIM4 remapping This bit is set and cleared by software. It controls the mapping of TIM4 channels 1 to 4 onto the GPIO ports. 0: No remap(CH1/PB6, CH2/PB7, CH3/PB8, CH4/PB9) 1: Full remap(CH1/PD12, CH2/PD13, CH3/PD14, CH4/PD15) **Note:** *TIM4_ETR on PE0 is not remapped.*
[11:10]	RW	**TIM3_REMAP[1:0]:** TIM3 remapping These bits are set and cleared by software, They control the mapping of TIM3 channels 1 to 4 on the GPIO ports. 00: No remap(CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1) 01: Not used 10: Partial remap(CH1/PB4, CH2/PB5, CH3/PB0, CH4/PB1) 11: Full remap(CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9) **Note:** *TIM3_ETR on PE0 is not remapped*
[9:8]	RW	**TIM2_REMAP[1:0]:** TIM2 remapping These bits are set and cleared by software. They control the mapping of TIM2 channels 1 to 4 and external trigger (ETR) on the GPIO ports. 00: No remap(CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3) 01: Partial remap(CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3) 10: Partial remap(CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11) 11: Full remap(CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11)

[7:6]	RW	**TIMx_REMAP[1:0]:** TIM1 remapping These bits are set and cleared by software. They control the mapping of TIM1 channels 1 to 4, 1N to 3N, external trigger (ETR) and Break input (BKIN) on the GPIO ports. 00: No remap(ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PB12, CH1N/PB13, CH2N/PB14, CH3N/PB15) 01: Partial remap(ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1) 10: not used 11: Full remap(ETR/PE7, CH1/PE9, CH2/PE11, CH3/PE13, CH4/PE14, BKIN/PE15, CH1N/PE8, CH2N/PE10, CH3N/PE12)
[5:4]	RW	**USART3_REMAP[1:0]:** USART3 remapping These bits are set and cleared by software. They control the mapping of USART3 CTS, RTS,CK,TX and RX alternate functions on the GPIO ports. 00: No remap(TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14) 01: Partial remap(TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14) 10: not used 11: Full remap(TX/PD8, RX/PD9, CK/PD10, CTS/PD11, RTS/PD12)
[3]	RW	**USART2_REMAP:** USART2 remapping This bit is set and cleared by software. It controls the mapping of USART2 CTS, RTS,CK,TX and RX alternate functions on the GPIO ports. 0: No remap(CTS/PA0, RTS/PA1, TX/PA2, RX/PA3, CK/PA4) 1: Remap(CTS/PD3, RTS/PD4, TX/PD5, RX/PD6, CK/PD7)
[2]	RW	**USART1_REMAP:** USART1 remapping This bit is set and cleared by software. It controls the mapping of USART1 TX and RX alternate functions on the GPIO ports. 0: No remap(TX/PA9, RX/PA10) 1: Remap(TX/PB6, RX/PB7)
[1]	RW	**I2C1_REMAP:** I2C1 remapping This bit is set and cleared by software. It controls the mapping of I2C1 SCL and SDA alternate functions on the GPIO ports. 0: No remap(SCL/PB6, SDA/PB7) 1: Remap(SCL/PB8, SDA/PB9)
[0]	RW	**SPI1_REMAP:** SPI1 remapping This bit is set and cleared by software. It controls the mapping of SPI1 NSS, SCK, MISO, MOSI alternate functions on the GPIO ports 0: No remap(NSS/PA4, SCK/PA5, MISO/PA6, MOSI/PA7) 1: Remap(NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5)

9.2.3 External interrupt configuration register 1 (AFIO_EXTICR1)

Register	Address offset	Access	Reset value	Description
AFIO_EXTICR1	0x08	RW	0x0000_0000	External interrupt configuration register 1

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

External interrupt configuration register 1(AFIO_EXTICR1)bit description

Bit	Access	Description
[31:16]	R	Reserved, Must remain clear
[15:0]	RW	**EXTIx[3:0]:** EXTI x configuration (x = 0 ~ 3) These bits are written by software to select the source input for EXTIx external interrupt. 0000: PA[x] 0001: PB[x] 0010: PC[x] 0011: PD[x] 0100: PE[x]

9.2.4 External interrupt configuration register 2 (AFIO_EXTICR2)

Register	Address offset	Access	Reset value	Description
AFIO_EXTICR2	0x0C	RW	0x0000_0000	External interrupt configuration register 2

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

External interrupt configuration register 2(AFIO_EXTICR2)bit description

Bit	Access	Description
[31:16]	R	Reserved, Must remain clear
[15:0]	RW	**EXTIx[3:0]:** EXTI x configuration (x = 4 ~ 7) These bits are written by software to select the source input for EXTIx external interrupt. 0000: PA[x] 0001: PB[x] 0010: PC[x] 0011: PD[x] 0100: PE[x]

9.2.5 External interrupt configuration register 3 (AFIO_EXTICR3)

Register	Address offset	Access	Reset value	Description
AFIO_EXTICR3	0x10	RW	0x0000_0000	External interrupt configuration register 3

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

External interrupt configuration register 3(AFIO_EXTICR3)bit description

Bit	Access	Description
[31:16]	R	Reserved, Must remain clear
[15:0]	RW	**EXTIx[3:0]:** EXTI x configuration (x = 8 ~ 11) These bits are written by software to select the source input for EXTIx external interrupt. 0000: PA[x] 0001: PB[x] 0010: PC[x] 0011: PD[x] 0100: PE[x]

9.2.6 External interrupt configuration register 4 (AFIO_EXTICR4)

Register	Address offset	Access	Reset value	Description
AFIO_EXTICR4	0x14	RW	0x0000_0000	External interrupt configuration register 4

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

External interrupt configuration register 4(AFIO_EXTICR3)bit description

Bit	Access	Description
[31:16]	R	Reserved, Must remain clear
[15:0]	RW	**EXTIx[3:0]:** EXTI x configuration (x = 12 ~ 15) These bits are written by software to select the source input for EXTIx external interrupt. 0000: PA[x] 0001: PB[x] 0010: PC[x] 0011: PD[x] 0100: PE[x]

10 Interrupts and events

10.1 Nested vectored interrupt controller(NVIC)

- 68 (not including the sixteen Cortex™-M3 interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

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

10.1.1 SysTick calibration value register

The SysTick calibration value is set to 9000, which gives a reference time base of 1 ms with the SysTick clock set to 9 MHz (max HCLK/8).

10.1.2 Interrupt and exception vectors

Tab 10.1-1 GPIO configurations for device peripherals

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000_0000
-	-3	fixed	Reset	Reset	0x0000_0004
-	-2	fixed	NMI	Nonmaskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
-	-1	fixed	HardFault	All class of fault	0x0000_000C
-	0	settable	MemManage	Memory management	0x0000_0010
-	1	settable	BusFault	Prefetch fault, memory access fault	0x0000_0014
-	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
-	-	-	-	Reserved	0x0000_001C - 0x0000_0028
-	3	settable	SVCall	System service call via SWI instruction	0x0000_002C
-	4	settable	Debug Monitor	Debug monitor	0x0000_0030
-	-	-	-	Reserved	0x0000_0034
-	5	settable	PendSV	Pendable request for system service	0x0000_0038
-	6	settable	SysTick	Systick timer	0x0000_003C
0	7	settable	WWDG	Window watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER	Tamper interrupt	0x0000_0048
3	10	settable	RTC	RTC global interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050

(Continuation)

Position	Priority	Type of priority	Acronym	Description	Address
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088
19	26	settable	USB_DMA_CAN	USB_DMA interrupt and CAN global interrupt	0x0000_008C
20	27	settable	USB	USB global interrupts	0x0000_0090
21	28	settable	-	Reserved	0x0000_0094
22	29	settable	-	Reserved	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	settable	TIM1_BRK	TIM1 Break interrupt	0x0000_00A0
25	32	settable	TIM1_UP	TIM1 Update interrupt	0x0000_00A4
26	33	settable	TIM1_TRG_COM	TIM1 Trigger and Commutation interrupts	0x0000_00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I2C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I2C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
42	49	settable	R	Reserved	0x0000_00E8
43	50	settable	TIM8_BRK	TIM8 Break interrupt	0x0000_00EC
44	51	settable	TIM8_UP	TIM8 Update	0x0000_00F0

(Continuation)

Position	Priority	Type of priority	Acronym	Description	Address
45	52	settable	TIM8_TRG_COM	TIM8 Trigger and Commutation interrupts	0x0000_00F4
46	53	settable	TIM8_CC	TIM8 Capture Compare interrupt	0x0000_00F8
47	54	settable	ADC3	ADC3 global interrupt	0x0000_00FC
48	55	settable	-	Reserved	0x0000_0100
49	56	settable	SDIO	SDIO global interrupt	0x0000_0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C
52	59	settable	UART4	UART4 global interrupt	0x0000_0110
53	60	settable	UART5	UART5 global interrupt	0x0000_0114
54	61	settable	TIM6	TIM6 global interrupt	0x0000_0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C
56	63	settable	DMA2_Channel1	DMA2 Channel1 global interrupt	0x0000_0120
57	64	settable	DMA2_Channel2	DMA2 Channel2 global interrupt	0x0000_0124
58	65	settable	DMA2_Channel3	DMA2 Channel3 global interrupt	0x0000_0128
59	66	settable	DMA2_Channel4_5	DMA2 Channel4 and DMA2 Channel5 global interrupts	0x0000_012C
60	67	settable	QSPI	DMA2 Channel5 global interrupts	0x0000_0130
61	68	settable	RNG	Random number generation module global interrupt	0x0000_0134
62	69	settable	AES	AES global interrupts	0x0000_0138
63	70	settable	USART_EXTI	5 UART wake up interrupts through EXTI	0x0000_013C

10.2 External interrupt/event controller (EXTI)

19 edge detectors in MG32F157xx for generating event/interrupt requests. Each input line can be independently configured to select the type (event or interrupt) and the corresponding trigger event (rising or falling or both). Each line can also masked independently. A pending register maintains the status line of the interrupt requests

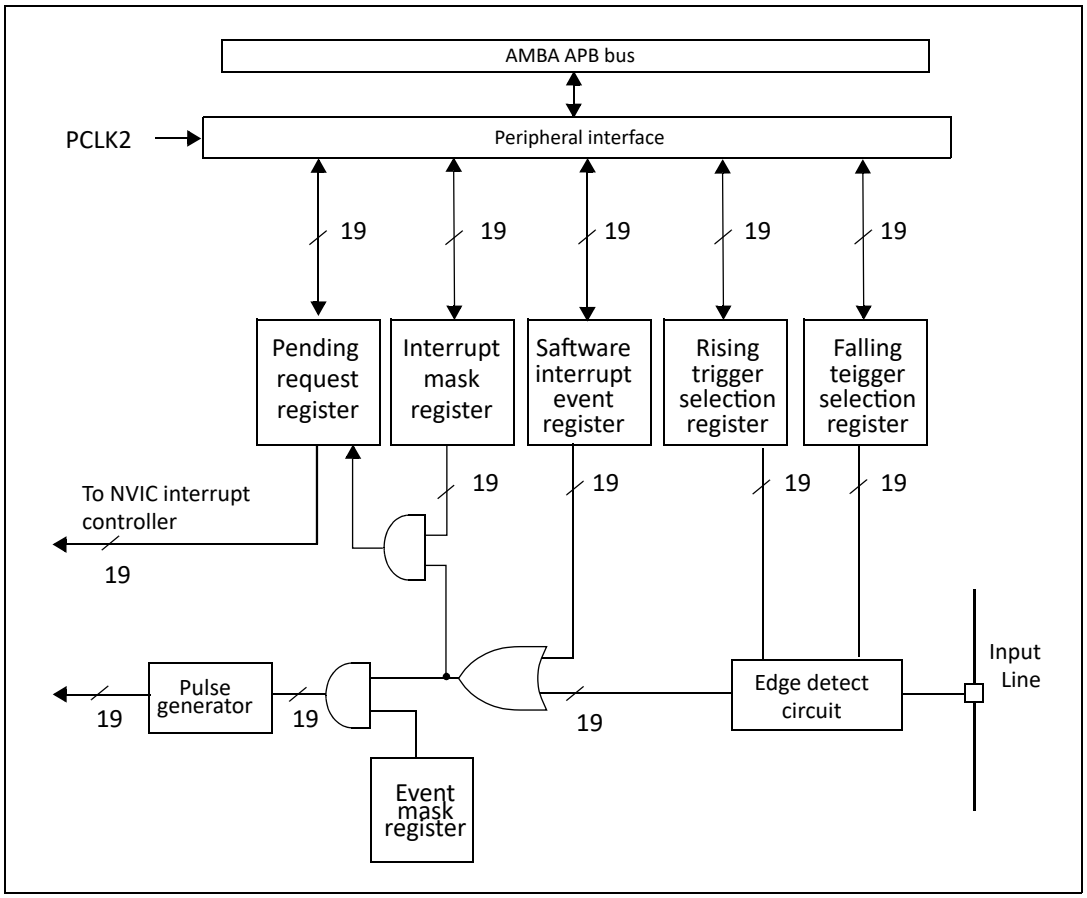
10.2.1 Main features

The EXTI controller main features are the following:

- Independent trigger and mask on each interrupt/event line
- Dedicated status bit for each interrupt line
- Generation of up to 20 software event/interrupt requests
- Detection of external signal with pulse width lower than APB2 clock period. Refer to the electrical characteristics section of the datasheet for details on this parameter.

10.2.2 Block diagram

Fig 10.2-1 External interrupt/event controller block diagram



MG32F157xx is able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated either by:

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

10.3 Generate interrupt

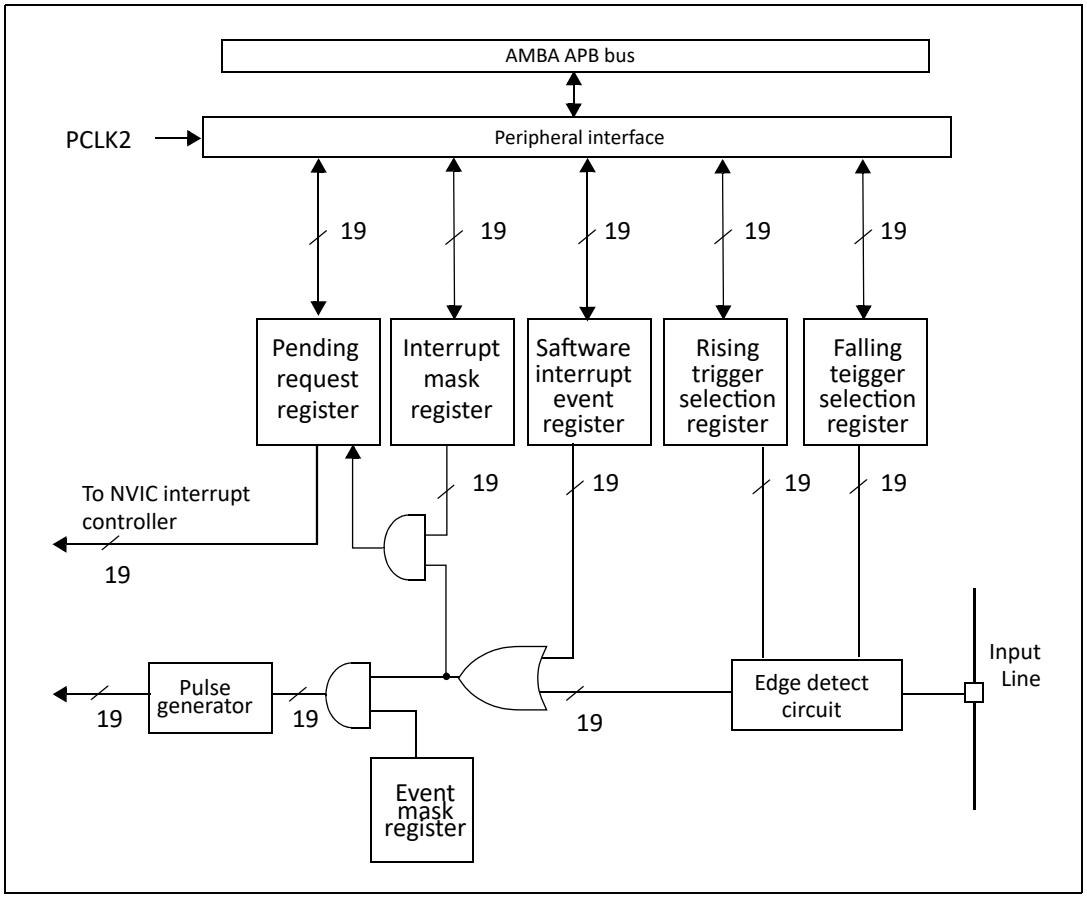
To generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1’ to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a ‘1’ in the pending register.

Hardware interrupt selection

To configure the 24 lines as interrupt sources, use the following procedure:

- Configure the mask bits of the 24 Interrupt linesEXTI_IMR)。
- Configure the Trigger Selection bits of the Interrupt lines（EXTI_RTSR 和 EXTI_FTSR)。
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the External Interrupt Controller (EXTI) so that an interrupt coming from one of the 24 lines can be correctly acknowledged.

Fig 10.3-1 External interrupt/event controller block diagram



10.3.1 Wakeup event management

MG32F157xx is able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated either by:

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

10.3.2 Functional description

To generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1’ to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a ‘1’ in the pending register.

To generate the event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a ‘1’ to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set

An interrupt/event request can also be generated by software by writing a ‘1’ in the software interrupt/event register.

Hardware interrupt selection

To configure the 20 lines as interrupt sources, use the following procedure:

- Configure the mask bits of the 20 Interrupt lines (EXTI_IMR)
- Configure the Trigger Selection bits of the Interrupt lines (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the External Interrupt Controller (EXTI) so that an interrupt coming from one of the 20 lines can be correctly acknowledged.

Hardware event selection

To configure the 20 lines as event sources, use the following procedure:

- Configure the mask bits of the 20 Event lines (EXTI_EMR)
- Configure the Trigger Selection bits of the Event lines (EXTI_RTSR and EXTI_FTSR)

Software interrupt/event selection

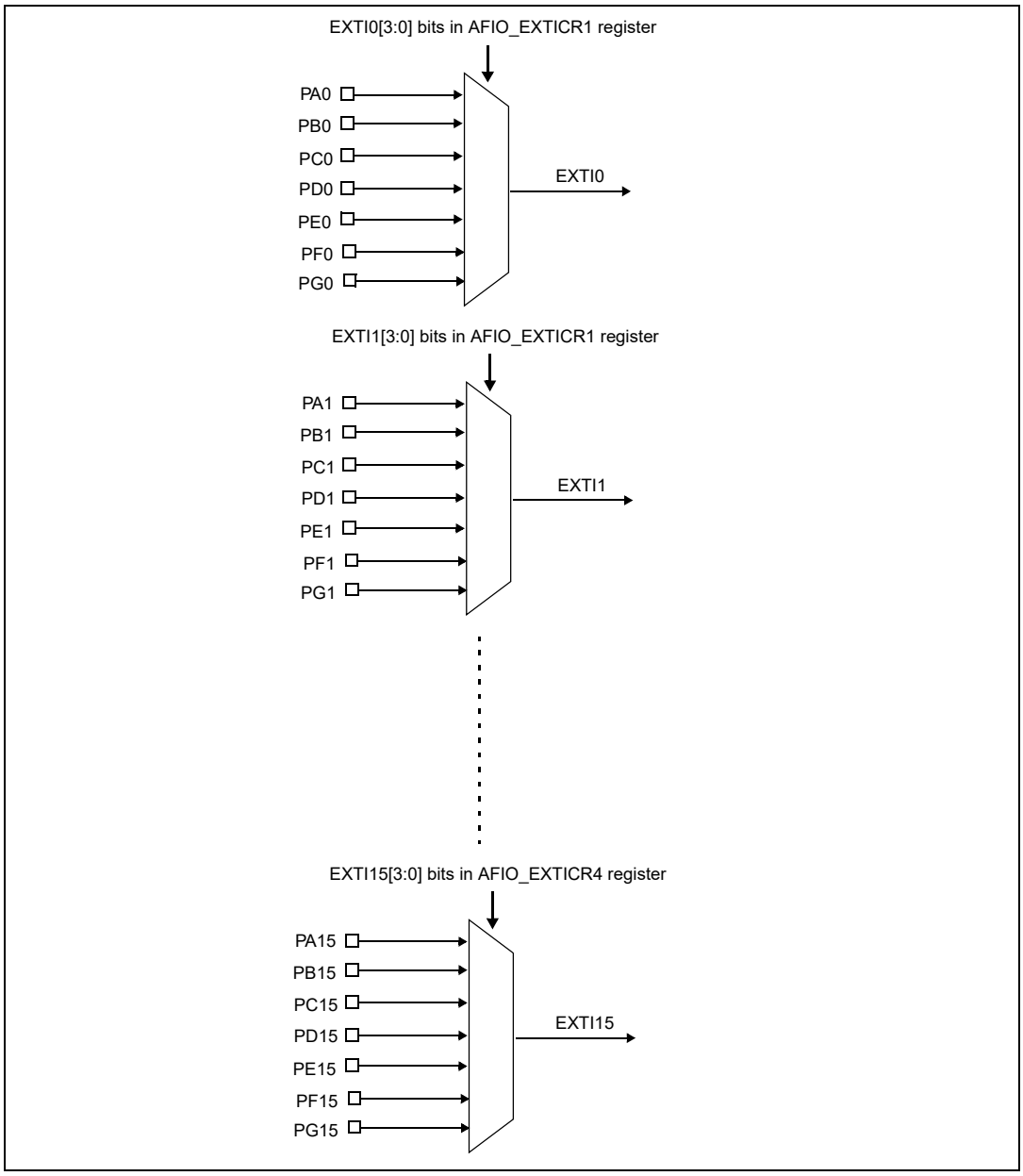
The 20 lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

- Configure the mask bits of the 20 Interrupt/Event lines (EXTI_IMR, EXTI_EMR)
- Set the required bit of the software interrupt register (EXTI_SWIER)

10.3.3 External interrupt/event line mapping

The 112 GPIOs are connected to the 16 external interrupt/event lines in the following manner:

Fig 10.3-2 External interrupt/event GPIO mapping



Note: *To configure the AFIO_EXTICRx for the mapping of external interrupt/event lines onto GPIOs, the AFIO clock should first be enabled.*

The two other EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event

10.4 EXTI Register

Tab 10.4-1 EXTI register map

offset	Register	Reset value	Description
EXTI: EXTI_BA = 0x4001_0400			
0x00	EXTI_IMR	0x0000_0000	Interrupt mask register
0x04	EXTI_EMR	0x0000_0000	Event mask register
0x08	EXTI_RTSR	0x0000_0000	Rising trigger selection register
0x0C	EXTI_FTSR	0x0000_0000	Falling trigger selection register
0x10	EXTI_SWIER	0x0000_0000	Software interrupt event register
0x14	EXTI_PR	0x0000_0000	Pending register

10.4.1 Interrupt mask register(EXTI_IMR)

Register	Address offset	Access	Reset value	Description
EXTI_IMR	0x00	RW	0x0000_0000	Interrupt mask register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					MR18	MR17	MR16
15	14	13	12	11	10	9	8
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8
7	6	5	4	3	2	1	0
MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0

Interrupt mask register(EXTI_IMR)bit description

Bit	Access	Description
[31:19]	R	Reserved, must be kept at reset value (0)
[18:0]	RW	**MRx[18:0]:** Interrupt Mask on line x 0: Interrupt request from Line x is masked 1: Interrupt request from Line x is not masked

10.4.2 Event mask register(EXTI_EMR)

Register	Address offset	Access	Reset value	Description
EXTI_EMR	0x04	RW	0x0000_0000	Event mask register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					MR18	MR17	MR16
15	14	13	12	11	10	9	8
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8
7	6	5	4	3	2	1	0
MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0

Event mask register(EXTI_EMR)bit description

Bit	Access	Description
[31:19]	R	Reserved, must remain clear
[18:0]	RW	**MRx[18:0]:** Event Mask on line x 0: Event request from Line x is masked 1: Event request from Line x is not masked

10.4.3 Rising trigger selection register(EXTI_RTSR)

Register	Address offset	Access	Reset value	Description
EXTI_RTSR	0x08	RW	0x0000_0000	Rising trigger selection register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					TR18	TR17	TR16
15	14	13	12	11	10	9	8
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8
7	6	5	4	3	2	1	0
TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0

Rising trigger selection register(EXTI_RTSR)bit description

Bit	Access	Description
[31:19]	R	Reserved, must be kept at reset value (0)
[18:0]	RW	**TRx[18:0]:** Rising trigger event configuration bit of line x 0: Rising trigger disabled (for Event and Interrupt) for input line 1: Rising trigger enabled (for Event and Interrupt) for input line

Note: *The external wakeup lines are edge triggered, no glitches must be generated on these lines. If a rising edge on external interrupt line occurs during writing of EXTI_RTSR register, the pending bit will not be set.Rising and Falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.*

10.4.4 Falling trigger selection register (EXTI_FTSR)

Register	Address offset	Access	Reset value	Description
EXTI_FTSR	0x0C	RW	0x0000_0000	Falling trigger selection register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					TR18	TR17	TR16
15	14	13	12	11	10	9	8
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8
7	6	5	4	3	2	1	0
TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0

Falling trigger selection register (EXTI_FTSR)bit description

Bit	Access	Description

[31:19]	R	Reserved, must be kept at reset value (0)
[18:0]	RW	**TRx[18:0]:** Falling trigger event configuration bit of line x 0: Falling trigger disabled (for Event and Interrupt) for input line 1: Falling trigger enabled (for Event and Interrupt) for input line.

Note: *The external wakeup lines are edge triggered, no glitches must be generated on these lines. If a falling edge on external interrupt line occurs during writing of EXTI_FTSR register, the pending bit will not be set.Rising and Falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.*

10.4.5 Software interrupt event register(EXTI_SWIER)

Register	Address offset	Access	Reset value	Description
EXTI_SWIER	0x10	RW	0x0000_0000	Software interrupt event register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					SWIER18	SWIER17	SWIER16
15	14	13	12	11	10	9	8
SWIER15	SWIER14	SWIER13	SWIER12	SWIER11	SWIER10	SWIER9	SWIER8
7	6	5	4	3	2	1	0
SWIER7	SWIER6	SWIER5	SWIER4	SWIER3	SWIER2	SWIER1	SWIER0

Software interrupt event register(EXTI_SWIER)bit description

Bit	Access	Description
[31:19]	R	Reserved, must be kept at reset value (0).
[18:0]	RW	**SWIERx[18:0]:** Software interrupt on line x If the interrupt is enabled on this line in the EXTI_IMR, writing a '1' to this bit when it is set to '0' sets the corresponding pending bit in EXTI_PR resulting in an interrupt request generation.This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a 1 into the bit).

10.4.6 Pending register(EXTI_PR)

Register	Address offset	Access	Reset value	Description
EXTI_PR	0x14	RW	0x0000_0000	Pending register

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

Pending register(EXTI_PR)bit description

Bit	Access	Description
[31:19]	R	Reserved，must be kept at reset value (0)
[18:0]	RC_W1	**PRx[18:0]:** Pending bit 0：No trigger request occurred 1：selected trigger request occurred This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a ‘1’ into the bit.

11 Analog-to-digital converter (ADC)

11.1 ADC introduction

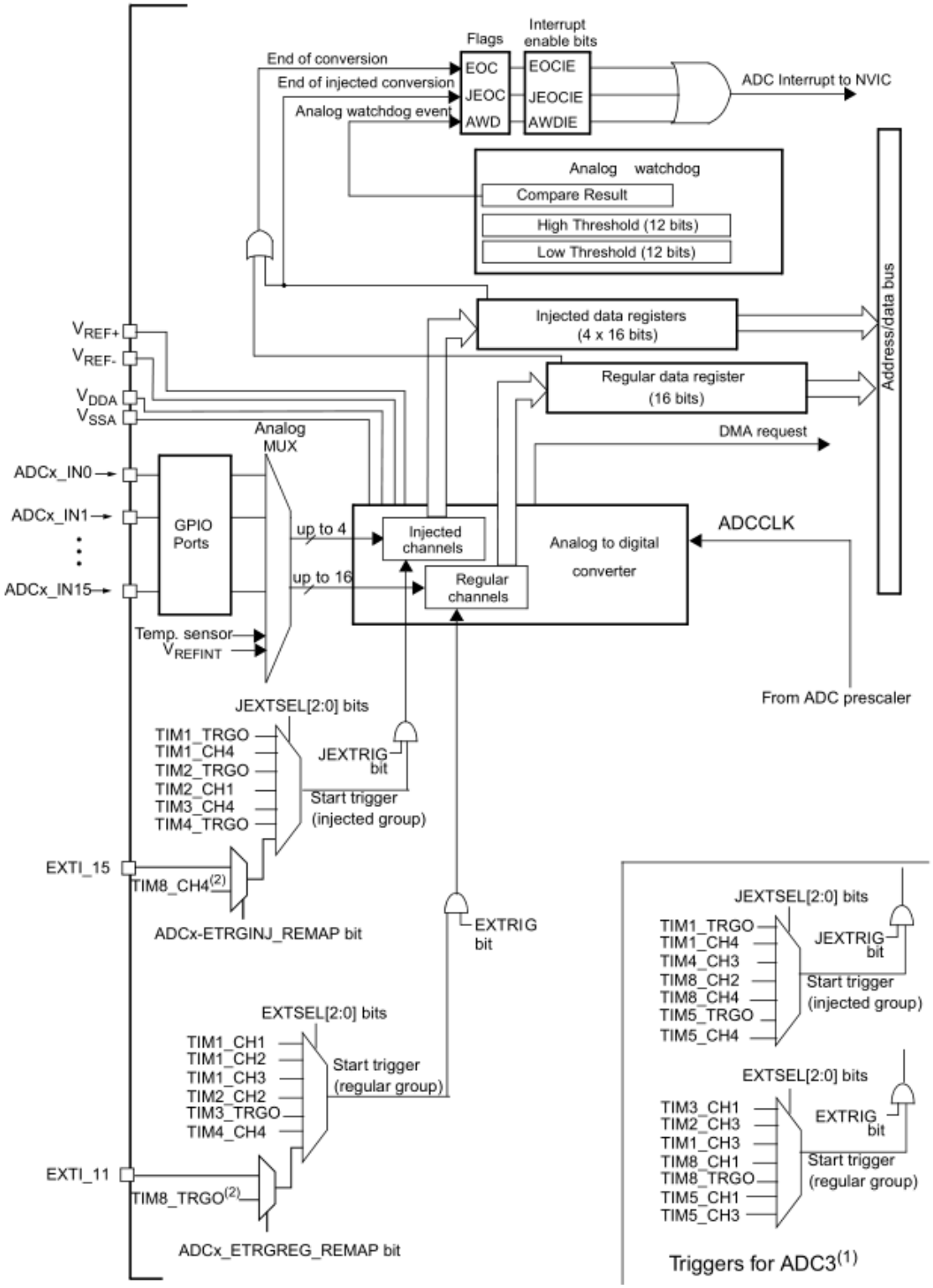
The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 18 multiplexed channels allowing it measure signals from sixteen external and two internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result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

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined high or low thresholds.The ADC input clock is generated from the PCLK2 clock divided by a prescaler and it must not exceed 16 MHz

11.2 ADC main features

- 12-bit resolution
- Interrupt generation at End of Conversion, End of Injected conversion and Analog watchdog event
- Single and continuous conversion modes
- Scan mode for automatic conversion of channel 0 to channel ‘n’
- Self-calibration
- Data alignment with in-built data coherency
- Channel by channel programmable sampling time
- External trigger option for both regular and injected conversion
- Discontinuous mode
- Dual mode (on devices with 2 ADCs or more)
- ADC conversion time
  - 1μs at 96MHz（The pre-division multiple is 6, the ADC clock is 16MHz, 4 sampling cycles plus 12 conversion cycles）
- ADC supply requirement: 2.4 V to 3.6 V
- ADC input range: $V_{REF-} \leqslant VIN \leqslant V_{REF+}$
- DMA request generation during regular channel conversion

Fig 11.2-1 ADC block diagram



1. ADC3 has regular and injected conversion triggers different from those of ADC1 and ADC2.

11.3 ADC Pins

Tab 11.3-1 ADC Pins Description

Name	Signal type	Remarks
VREF+	Input, analog reference positive	higher/positive reference voltage for ADC, $2.4V \leqslant V_{REF+} \leqslant V_{DDA}$
VREF-	Input, analog reference negative	lower/negative reference voltage for ADC, $V_{REF-} = V_{SSA}$
VDDA	Input, analog supply	Analog supply, $2.4V \leqslant V_{DDA} = V_{DD} \leqslant 3.6V$
VSSA	Input, analog supply ground	Analog supply ground, $V_{SSA} = V_{SS}$
ADCx_IN[15:0]	Analog signals	Up to 21 analog channels

[1] VDDA and VSSA have to be connected to VDD and VSS , respectively.

11.4 ADC Function Description

11.4.1 ADC on-off control

The ADC can be powered-on by setting the ADON bit in the ADC_CR2 register. When the ADON bit is set for the first time, it wakes up the ADC from Power Down mode.Conversion starts when ADON bit is set for a second time by software after ADC power-up time (t_{STAB}).

The conversion can be stopped, and the ADC put in power down mode by resetting the ADON bit. In this mode the ADC consumes almost no power (only a few μA).

11.4.2 ADC clock

The ADCCLK clock provided by the Clock Controller is synchronous with the PCLK2 (APB2 clock). The RCC controller has a dedicated programmable prescaler for the ADC clock, refer to Section7

11.4.3 Channel selection

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

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

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

Temperature sensor / V_{REFINT} internal channels

The Temperature sensor is connected to channel ADCx_IN16 and the internal reference voltage V_{REFINT} is connected to ADCx_IN17. These two internal channels can be selected and converted as injected or regular channels. **Note:** *The sensor and V_{REFINT} are only available on the master ADC1 peripheral.*

11.4.4 Single conversion mode

In Single conversion mode the ADC does one conversion. This mode is started either by setting the ADON bit in the ADC_CR2 register (for a regular channel only) or by external trigger (for a regular or injected channel), while the CONT bit is 0.
Once the conversion of the selected channel is complete:

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

The ADC is then stopped.

11.4.5 Continuous conversion mode

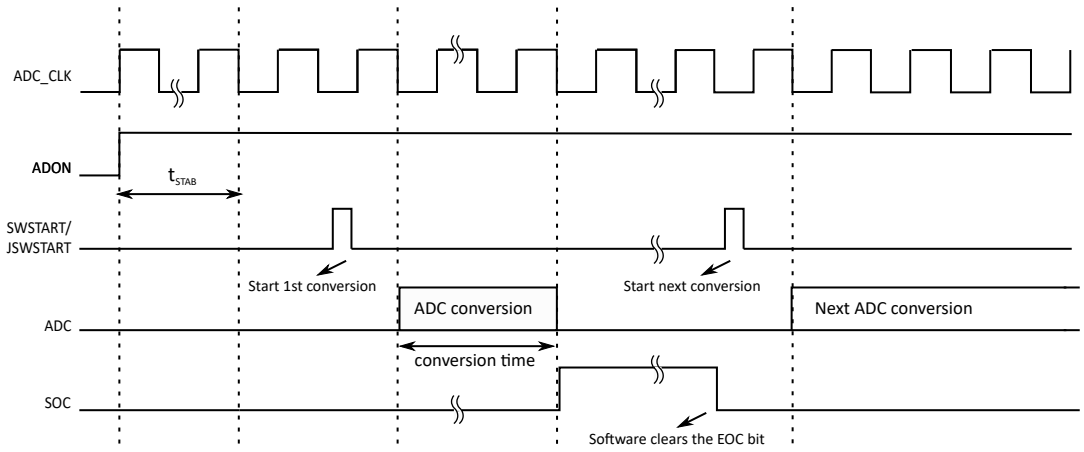
In continuous conversion mode ADC starts another conversion as soon as it finishes one. This mode is started either by external trigger or by setting the ADON bit in the ADC_CR2 register, while the CONT bit is 1.
After each conversion:

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

11.4.6 ADC Timing diagram

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

Fig 11.4-1 ADC Timing diagram



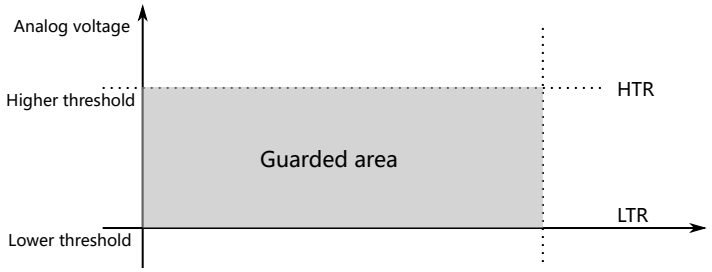
11.4.7 Analog watchdog

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

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

The analog watchdog can be enabled on one or more channels by configuring the ADC_CR1 register

Fig 11.4-2 Analog watchdog guarded area



Tab 11.4-1 Analog watchdog channel selection

Channels to be guarded by analog watchdog	ADC_CR1 register control bits		
	AWDSGL	AWDEN	JAWDEN
None	Arbitrary	0	1
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single[1] injected channel	1	0	1
Single[1] regular channel	1	1	0
Single[1] regular or injected channel	1	1	1

[1] Selected by AWDCH[4:0] bits

11.4.8 Scan mode

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

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

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

11.4.9 Injected channel management

Triggered injection

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

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

Note: *When using triggered injection, the interval between trigger events must be longer than the injection sequence. For instance, if the sequence length is 16 ADC clock cycles (That is 4 sampling clock cycles plus 12 conversion cycles), the minimum interval between triggers must be 17 ADC clock cycles.*

Auto-injection

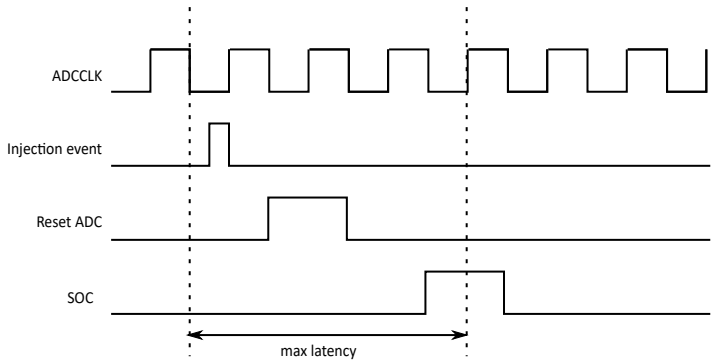
If the JAUTO bit is set, then the injected group channels are automatically converted after the regular group channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRx and ADC_JSQR registers.

In this mode, external trigger on injected channels must be disabled. If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

For ADC clock prescalers ranging from 4 to 8, a delay of 1 ADC clock period is automatically inserted when switching from regular to injected sequence (respectively injected to regular). When the ADC clock prescaler is set to 2, the delay is 2 ADC clock periods.

Note: *It is not possible to use both auto-injected and discontinuous modes simultaneously.*

Fig 11.4-3 Injected conversion latency



Note: *The maximum latency value can be found in the electrical characteristics of the datasheets.*

11.4.10 Discontinuous mode

Regular group

This mode is enabled by setting the DISCEN bit in the ADC_CR1 register. It can be used to convert a short sequence of n conversions (n <=8) which is a part of the sequence of conversions selected in the ADC_SQRx registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRx registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example: n = 3, channels to be converted = 0, 1, 2, 3, 6, 7, 9, 10

- first trigger: sequence converted 0, 1, 2. An EOC event is generated at each conversion
- second trigger: sequence converted 3, 6, 7. An EOC event is generated at each conversion
- third trigger: sequence converted 9, 10. An EOC event is generated at each conversion
- fourth trigger: sequence converted 0, 1, 2. An EOC event is generated at each conversion

Note:

When a regular group is converted in discontinuous mode, no rollover will occur. When all sub groups are converted, the next trigger starts conversion of the first sub-group.In the example above, the fourth trigger reconverts the first sub-group channels 0, 1 and 2.

Injected group

This mode is enabled by setting the JDISCEN bit in the ADC_CR1 register. It can be used to convert the sequence selected in the ADC_JSQR register, channel by channel, after an external trigger event.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:n = 1, channels to be converted = 1, 2, 3

- first trigger: channel 1 converted and JEOC events generated
- second trigger: channel 2 converted and JEOC events generated
- third trigger: channel 3 converted and JEOC events generated
- fourth trigger: channel 1 and JEOC events generated

Note:

1. When all injected channels are converted, the next trigger starts the conversion of the first injected channel. In the example above, the fourth trigger reconverts the first injected channel 1.

2. It is not possible to use both auto-injected and discontinuous modes simultaneously.

3. The user must avoid setting discontinuous mode for both regular and injected groups together. Discontinuous mode must be enabled only for one group conversion.

11.5 Data alignment

ALIGN bit in the ADC_CR2 register selects the alignment of data stored after conversion. Data can be left or right aligned as shown in Tab:11.5-2 and Tab:11.5-1.

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

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

Tab 11.5-1 Right alignment of data

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

Tab 11.5-2 Left alignment of data

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

11.6 Channel-by-channel programmable sample time

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

The total conversion time is calculated as follows:

T_CONV = Sampling time + 12 cycles

Example:

F_{ADC_CLK} = 16MHz, and a sampling time of 4 cycles,

T_{CONV}= 4 + 12 = 16 cycles = 1μs

11.7 Conversion on external trigger

Conversion can be triggered by an external event (e.g. timer capture, EXTI line). If the EXT- TRIG control bit is set then external events are able to trigger a conversion. The EXT- SEL[2:0] and JEXTSEL[2:0] control bits allow the application to select decide which out of 8 possible events can trigger conversion for the regular and injected groups.

Note: *When an external trigger is selected for ADC regular or injected conversion, only the rising edge of the signal can start the conversion.*

Tab 11.7-1 External trigger for regular channels for ADC1 and ADC2

Source	Type	EXTSEL[2:0]
TIM1_CC1 event	Internal signal from on-chip timers	000
TIM1_CC2 event		001
TIM1_CC3 event		010
TIM2_CC2 event		011
TIM3_TRGO event		100
TIM4_CC4 event		101
EXTI line11/TIM8_TRGOevent[1]	External pin/Internal signal from on-chip timers	110
SWSTART	Software control bit	111

Note: *The selection of the external trigger EXTI line11 or TIM8_TRGO event for regular channels is done through configuration bits ADC1_ETRGREG_REMAP and ADC2_ETRGREG_REMAP for ADC1 and ADC2, respectively.*

Tab 11.7-2 External trigger for injected channels for ADC1 and ADC2

Source	Type	JEXTSEL[2:0]
TIM1_TRGO event	Internal signal from on-chip timers	000
TIM1_CC4 event		001
TIM2_TRGO event		010
TIM2_CC1 event		011
TIM3_CC4 event		100
TIM4_TRGO event		101
EXTI line 15/TIM8_CC4 event[1]	External pin/Internal signal from on-chip timers	110
JSWSTART	Software control bit	111

Note: *The selection of the external trigger EXTI line15 or TIM8_CC4 event for injected channels is done through configuration bits ADC1_ETRGINJ_REMAP and ADC2_ETRGINJ_REMAP for ADC1 and ADC2, respectively.*

Tab 11.7-3 External trigger for regular channels for ADC3

Source	Type	EXTSEL[2:0]
TIM3_CC1 event	Internal signal from on-chip timers	000
TIM2_CC3 event		001
TIM1_CC3 event		010
TIM8_CC1 event		011
TIM8_TRGO event		100
TIM5_CC1 event		101
TIM5_CC3 event		110
SWSTART	Software control bit	111

Tab 11.7-4 External trigger for injected channels for ADC3

Source	Type	JEXTSEL[2:0]
TIM1_TRGO event	Internal signal from on-chip timers	000
TIM1_CC4 event		001
TIM4_CC3 event		010
TIM8_CC2 event		011
TIM8_CC4 event		100
TIM5_TRGO event		101
TIM5_CC4 event		110
JSWSTART	Software control bit	111

The software source trigger events can be generated by setting a bit in a register(SWSTART and JSWSTART in ADC_CR2). A regular group conversion can be interrupted by an injected trigger.

11.8 DMA request

Since converted regular channels value are stored in a unique data register, it is necessary to use DMA for conversion of more than one regular channel. This avoids the loss of data already stored in the ADC_DR register. Only the end of conversion of a regular channel generates a DMA request, which allows the transfer of its converted data from the ADC_DR register to the destination location selected by the user.

Note: *Only ADC1 and ADC3 have this DMA capability. ADC2-converted data can be transferred in dual ADC mode using DMA thanks to master ADC1.*

11.9 Dual ADC mode

In devices with two ADCs or more, dual ADC mode can be used (see Figure 11.9-1).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADC1 master to the ADC2 slave, depending on the mode selected by the DUALMOD[3:0] bits in the ADC1_CR1 register.

Note: *In dual mode, when configuring conversion to be triggered by an external event, the user must set the trigger for the master only and set a software trigger for the slave to prevent spurious triggers to start unwanted slave conversion. However,*

external triggers must be enabled on both master and slave ADCs.

The following six possible modes are implemented:

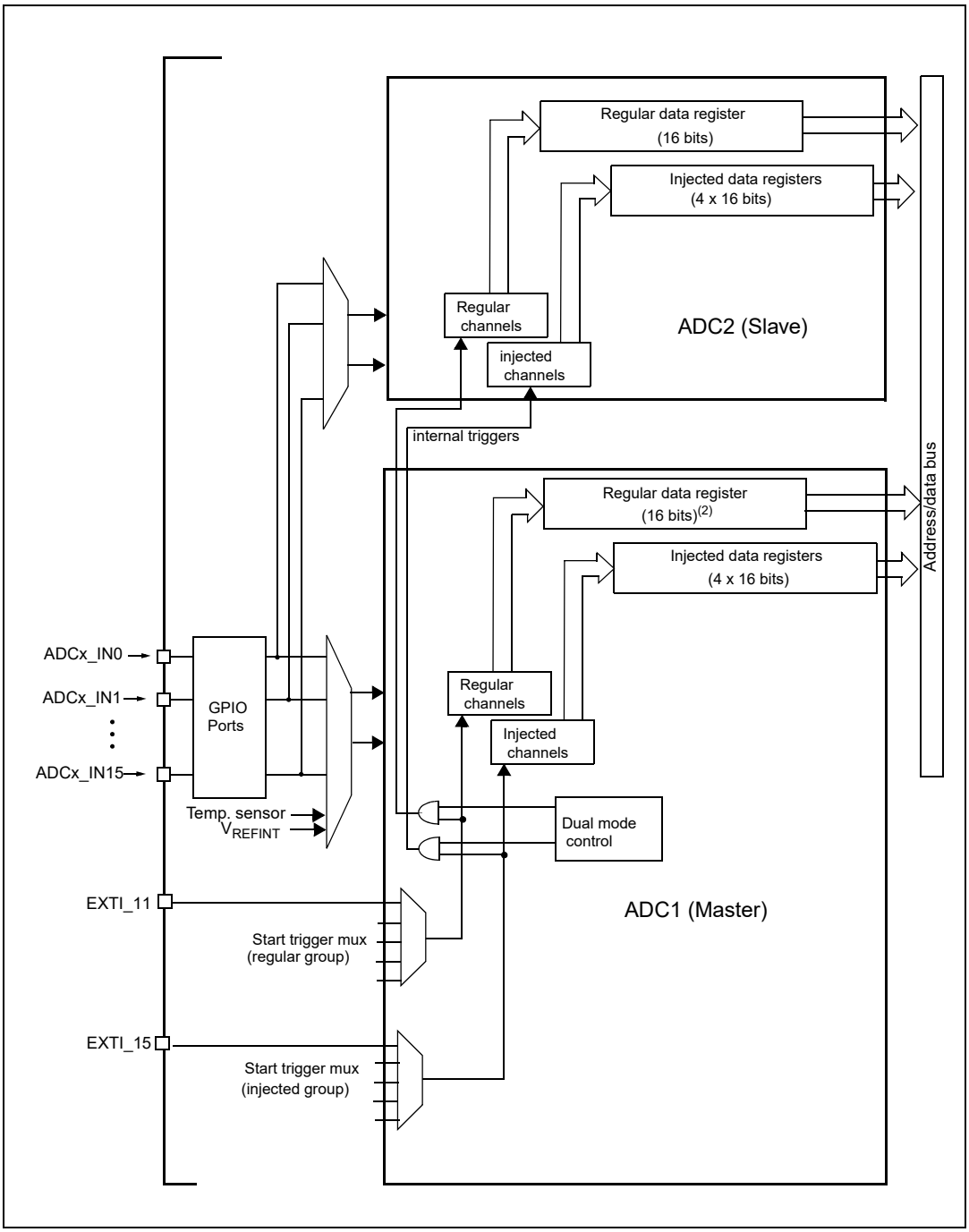
- Injected simultaneous mode
- Regular simultaneous mode
- Fast interleaved mode
- Slow interleaved mode
- Alternate trigger mode
- Independent mode

It is also possible to use the previous modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode
- Injected simultaneous mode + Interleaved mode

Note: *In dual ADC mode, to read the slave converted data on the master data register, the DMA bit must be enabled even if it is not used to transfer converted regular channel data.*

Fig 11.9-1 Dual ADC block diagram



1. External triggers are present on ADC2 but are not shown for the purposes of this diagram.
2. In some dual ADC modes, the ADC1 data register (ADC1_DR) contains both ADC1 and ADC2 regular converted data over the entire 32 bits.

11.9.1 Injected simultaneous mode

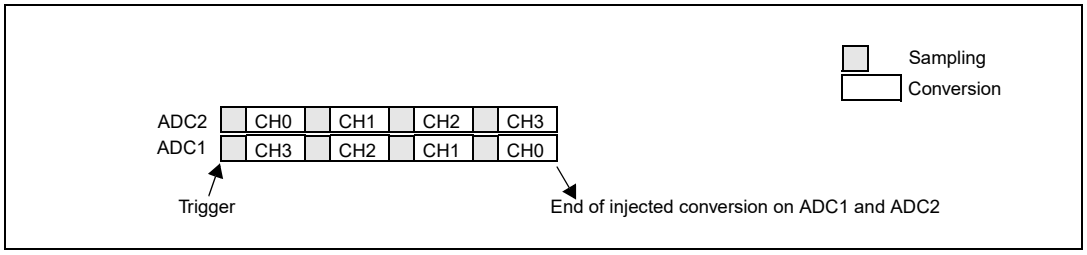
This mode converts an injected channel group. The source of external trigger comes from the injected group mux of ADC1 (selected by the JEXTSEL[2:0] bits in the ADC1_CR2 register). A simultaneous trigger is provided to ADC2. **Note:** *Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).*

At the end of conversion event on ADC1 or ADC2:

- The converted data is stored in the ADC_JDRx registers of each ADC interface.
- An JEOC interrupt is generated (if enabled on one of the two ADC interfaces) when the ADC1/ADC2 injected channels are all converted.

Note: *In Injected simultaneous mode, the conversion sequence must be the same length, or the trigger interval must be longer than the longer sequence of the two sequences. Otherwise, the conversion of the shorter sequence may be restarted when the conversion of the longer sequence is not completed.*

Fig 11.9-2 Injected simultaneous mode on 4 channels



11.9.2 Regular simultaneous mode

his mode is performed on a regular channel group. The source of the external trigger comes from the regular group mux of ADC1 (selected by the EXTSEL[2:0] bits in the ADC1_CR2 register). A simultaneous trigger is provided to the ADC2.

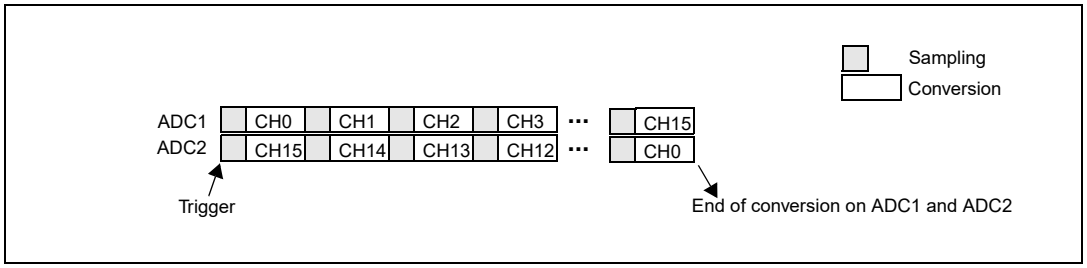
Note: *Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).*

At the end of conversion event on ADC1 or ADC2:

- A 32-bit DMA transfer request is generated (if DMA bit is set) which transfers to SRAM the ADC1_DR 32-bit register containing the ADC2 converted data in the upper halfword and the ADC1 converted data in the lower halfword.
- An EOC interrupt is generated (if enabled on one of the two ADC interfaces) when ADC1/ADC2 regular channels are all converted.

Note: *In Regular simultaneous mode, the conversion sequence must be the same length, or the trigger interval must be longer than the longer sequence of the two sequences. Otherwise, the conversion of the shorter sequence may be restarted when the conversion of the longer sequence is not completed.*

Fig 11.9-3 Regular simultaneous mode on 16 channels



11.9.3 Fast interleaved mode

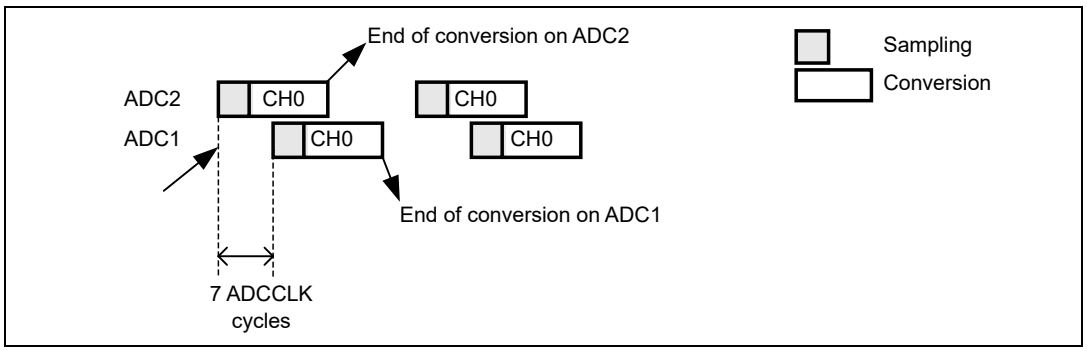
This mode can be started only on a regular channel group (usually one channel). The source of external trigger comes from the regular channel mux of ADC1. After an external trigger occurs:

- ADC2 starts immediately
- ADC1 starts after a delay of 7 ADC clock cycles

After an EOC interrupt is generated by ADC1 (if enabled through the EOCIE bit) a 32-bit DMA transfer request is generated (if the DMA bit is set) which transfers to SRAM the ADC1_DR 32-bit register containing the ADC2 converted data in the upper halfword and the ADC1 converted data in the lower halfword.

Note: *The maximum sampling time allowed is <7 ADCCLK cycles to avoid the overlap sampling phases*

Fig 11.9-4 Fast interleaved mode on 1 channel in continuous conversion mode



11.9.4 Slow interleaved mode

This mode can be started only on a regular channel group (only one channel). The source of external trigger comes from regular channel mux of ADC1. After external trigger occurs:

- ADC2 starts immediately
- ADC1 starts after a delay of 14 ADC clock cycles
- ADC2 starts after a second delay of 14 ADC cycles, and so on

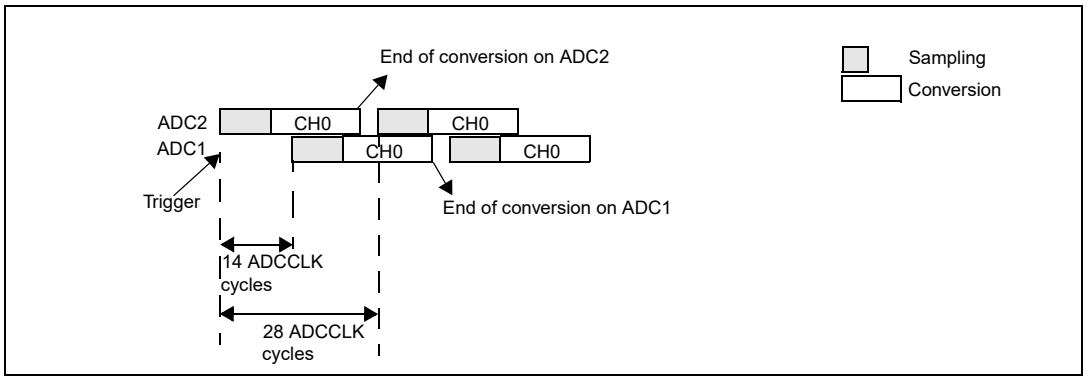
After an EOC interrupt is generated by ADC1 (if enabled through the EOCIE bit) a 32-bit DMA transfer request is generated (if the DMA bit is set) which transfers to SRAM the ADC1_DR 32-bit register containing the ADC2 converted data in the upper halfword and the ADC1 converted data in the lower halfword.

A new ADC2 start is automatically generated after 28 ADC clock cycles

CONT bit can not be set in the mode since it continuously converts the selected regular channel.

Note: *The maximum sampling time allowed is <14 ADCCLK cycles to avoid an overlap with the next conversion.The application must ensure that no external trigger for injected channel occurs when interleaved mode is enabled.*

Fig 11.9-5 Slow interleaved mode on 1 channel



11.9.5 Alternate trigger mode

This mode can be started only on an injected channel group. The source of external trigger comes from the injected group mux of ADC1.

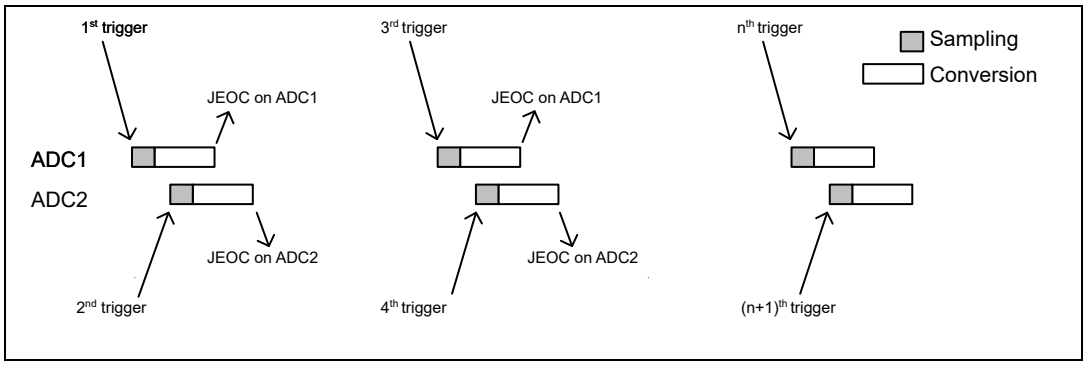
- When the first trigger occurs, all injected group channels in ADC1 are converted.
- When the second trigger arrives, all injected group channels in ADC2 are converted
- and so on.

A JEOC interrupt, if enabled, is generated after all injected group channels of ADC1 are converted.

A JEOC interrupt, if enabled, is generated after all injected group channels of ADC2 are converted.

If another external trigger occurs after all injected group channels have been converted then the alternate trigger process restarts by converting ADC1 injected group channels.

Fig 11.9-6 Alternate trigger: injected channel group of each ADC



If the injected discontinuous mode is enabled for both ADC1 and ADC2:

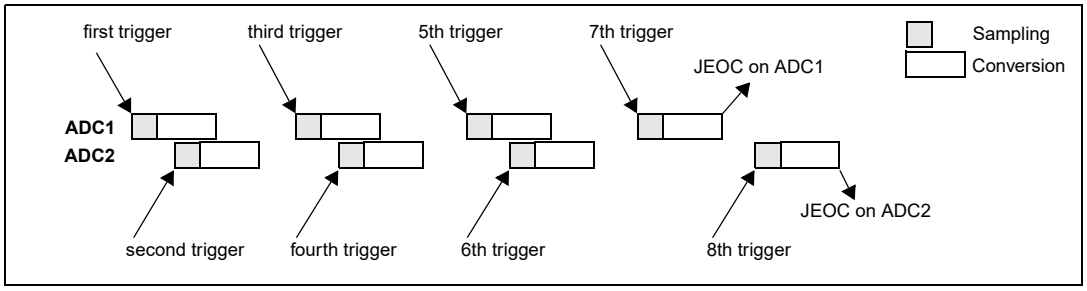
- When the first trigger occurs, the first injected channel in ADC1 is converted
- When the second trigger arrives, the first injected channel in ADC2 are converted
- and so on...

A JEOC interrupt, if enabled, is generated after all injected group channels of ADC1 are converted.

A JEOC interrupt, if enabled, is generated after all injected group channels of ADC2 are converted.

If another external trigger occurs after all injected group channels have been converted then the alternate trigger process restarts.

Fig 11.9-7 Alternate trigger: injected channels (each ADC) in discontinuous model



11.9.6 Independent mode

In this mode the dual ADC synchronization is bypassed and each ADC interfaces works independently.

11.9.7 Combined regular/injected simultaneous mode

It is possible to interrupt simultaneous conversion of a regular group to start simultaneous conversion of an injected group.

Note: *In combined regular/injected simultaneous mode, exactly the same sampling time should be configured for the two channels that will be sampled simultaneously by ACD1 and ADC2.*

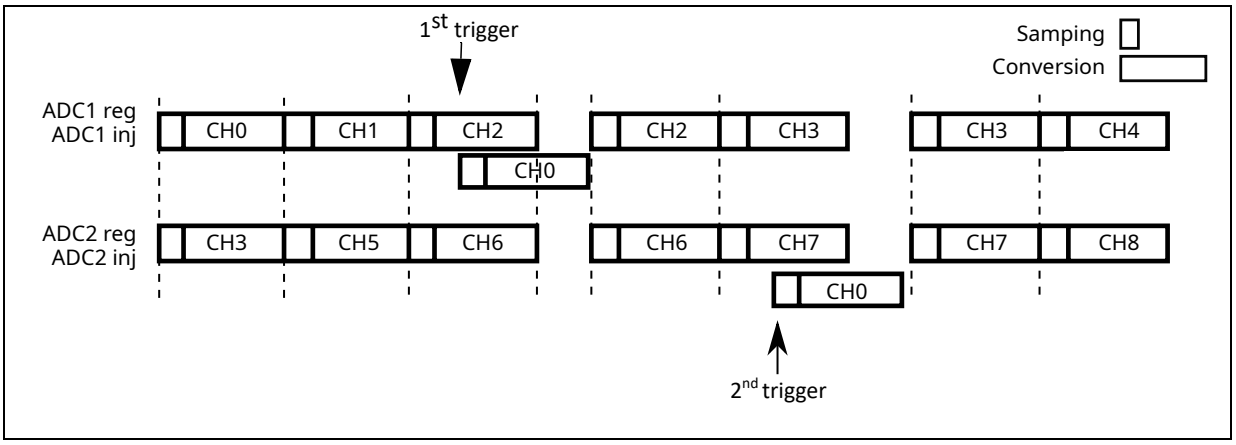
11.9.8 Combined regular simultaneous + alternate trigger mode

It is possible to interrupt regular group simultaneous conversion to start alternate trigger conversion of an injected group.

The injected alternate conversion is immediately started after the injected event arrives. If regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of both (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

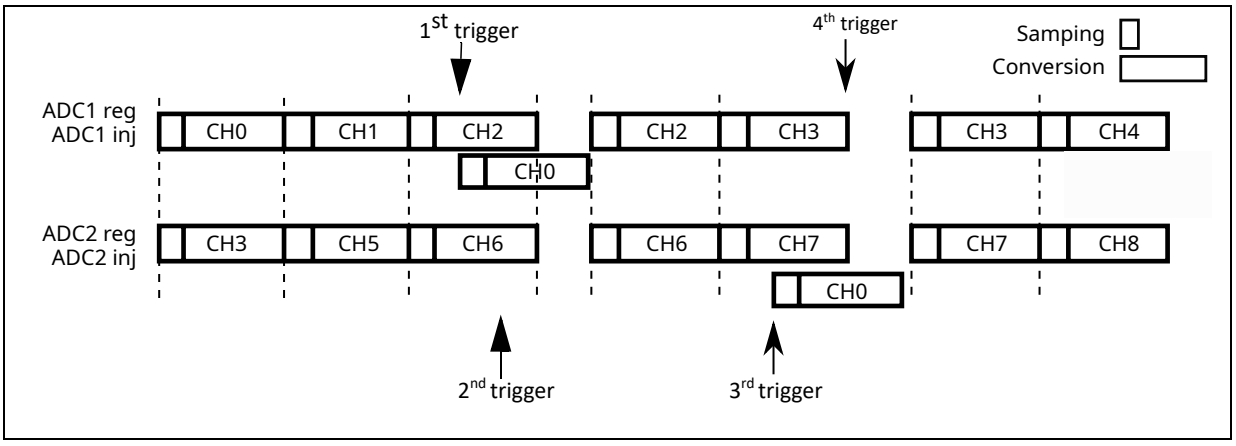
Note: *In combined regular simultaneous + alternate trigger mode, exactly the same sampling time should be configured for the two channels.Or ensure that the firing interval is longer than the longer of the two sequences, otherwise the ADC conversion with the shorter sequence may be restarted when the conversion of the longer sequence is not complete*

Fig 11.9-8 Alternate + Regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, it will be ignored.

Fig 11.9-9 Case of trigger occurring during injected conversion

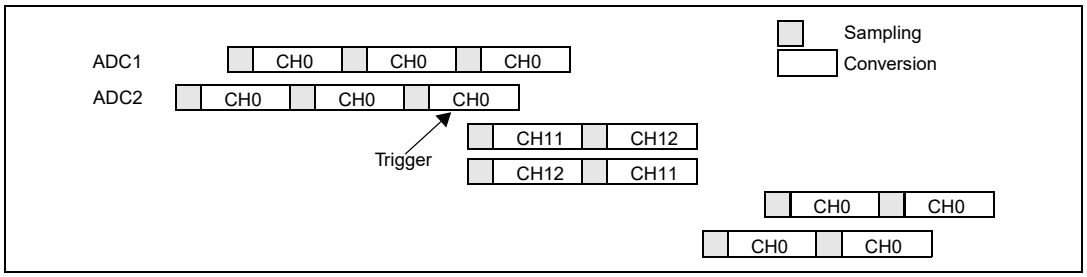


11.9.9 Combined injected simultaneous + interleaved

It is possible to interrupt an interleaved conversion with an injected event. In this case the interleaved conversion is interrupted and the injected conversion starts, at the end of the injected sequence the interleaved conversion is resumed.

Note: *When the ADC clock prescaler is set to 4, the interleaved mode does not recover with evenly spaced sampling periods: the sampling interval is 8 ADC clock periods followed by 6 ADC clock periods, instead of 7 clock periods followed by 7 clock periods.*

Fig 11.9-10 Interleaved single channel interrupted by injected sequence



11.9.10 Temperature sensor

The temperature sensor can be used to measure the ambient temperature (T_A) of the device.

The temperature sensor is internally connected to the ADCx_IN16 input channel which is used to convert the sensor output voltage into a digital value. The recommended sampling time for the temperature sensor is 17.1 µs.

The block diagram of the temperature sensor is shown in Figure 11.9-11.

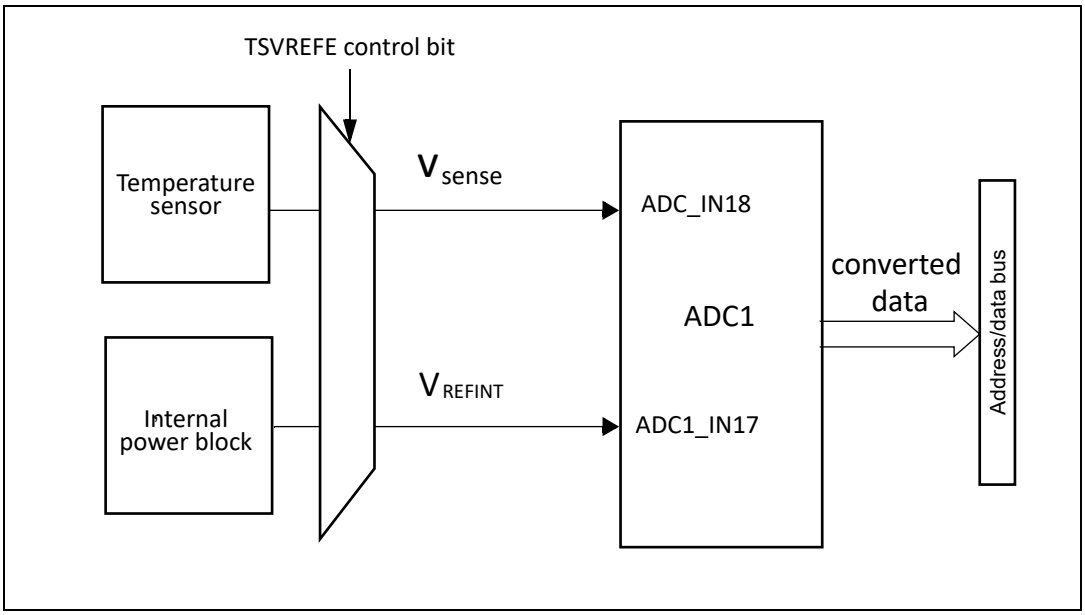
When not in use, this sensor can be put in power down mode

Note: *The TSVREFE bit must be set to enable both internal channels: ADCx_IN16 (temperature sensor) and ADCx_IN17 (*V_REFINT*) conversion.*

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The internal temperature sensor is more suited to applications that detect temperature variations instead of absolute temperatures. If accurate temperature readings are needed, an external temperature sensor part should be used.

Fig 11.9-11 Temperature sensor and V_{REFINT} channel block diagram



Reading the temperature

1. Select the ADCx_IN16 input channel.
2. Select a sample time of 17.1 μs
3. Set the TSVREFE bit in the ADC control register 2 (ADC_CR2) to wake up the temperature sensor from power down mode.
4. Start the ADC conversion by setting the ADON bit (or by external trigger).
5. Read the resulting V_{SENSE} data in the ADC data register
6. Obtain the temperature using the following formula:
   Temperature (in °C) = (V_{25} - V_{SENSE}) / Avg_Slope + 25.
   Where,
   V_{25} = V_{SENSE} value for 25° C and
   Avg_Slope = Average Slope for curve between Temperature vs. V_{SENSE} (given in mV/° C or μV/ °C).
   Refer to the Electrical characteristics section for the actual values of V_{25} and Avg_Slope.

Note: *The sensor has a startup time after waking from power down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADON and TSVREFE bits should be set at the same time.*

11.10 ADC interrupts

An interrupt can be produced on end of conversion for regular and injected groups and when the analog watchdog status bit is set. Separate interrupt enable bits are available for flexibility.

Note: *ADC1 and ADC2 interrupts are mapped onto the same interrupt vector. ADC3 interrupts are mapped onto a separate interrupt vector.*

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

- JSTRT (Start of conversion for injected group channels)
- STRT (Start of conversion for regular group channels)

Tab 11.10-1 ADC interrupts

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

11.11 ADC Register

Tab 11.11-1 ADC registers map

offset	Register	Reset value	Description
ADC1: ADC1_BA = 0x4001_2400 ADC2: ADC2_BA = 0x4001_2800			
0x00	ADC_SR	0x0000_0000	ADC status register
0x04	ADC_CR1	0x0000_0000	ADC control register 1
0x08	ADC_CR2	0x0000_0000	ADC control register 2
0x0C	ADC_SMPR1	0x0000_0000	ADC sample time register 1
0x10	ADC_SMPR2	0x0000_0000	ADC sample time register 2
0x14	ADC_JOFR1	0x0000_0000	ADC injected channel data offset register 1
0x18	ADC_JOFR2	0x0000_0000	ADC injected channel data offset register 2
0x1C	ADC_JOFR3	0x0000_0000	ADC injected channel data offset register 3
0x20	ADC_JOFR4	0x0000_0000	ADC injected channel data offset register 4
0x24	ADC_HTR	0x0000_0000	ADC watchdog high threshold register
0x28	ADC_LTR	0x0000_0000	ADC watchdog low threshold register
0x2C	ADC_SQR1	0x0000_0000	ADC regular sequence register 1
0x30	ADC_SQR2	0x0000_0000	ADC regular sequence register 2
0x34	ADC_SQR3	0x0000_0000	ADC regular sequence register 3
0x38	ADC_JSQR	0x0000_0000	ADC injected sequence register
0x3C	ADC_JDR1	0x0000_0000	ADC injected data register 1
0x3C	ADC_JDR2	0x0000_0000	ADC injected data register 2
0x44	ADC_JDR3	0x0000_0000	ADC injected data register 3
0x48	ADC_JDR4	0x0000_0000	ADC injected data register 4
0x4C	ADC_DR	0x0000_0000	ADC regular data register

11.11.1 ADC status register (ADC_SR)

Register	Address offset	Access	Reset value	Description
ADC_SR	0x00	RW	0x0000_0000	ADC status register

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

ADC status register (ADC_SR)bit description

Bit	Access	Description
[31:5]	R	Reserved, must be kept at 0。
[4]	RC_W0	**STRT:** Regular channel Start flag This bit is set by hardware when regular channel conversion starts. It is cleared by software. 0: No regular channel conversion started 1: Regular channel conversion has started
[3]	RC_W0	**JSTRT:** Injected channel Start flag This bit is set by hardware when injected channel group conversion starts. It is cleared by software. 0: No injected group conversion started 1: Injected group conversion has started
[2]	RC_W0	**JEOC:**Injected channel end of conversion This bit is set by hardware at the end of all injected group channel conversion. It is cleared by software. 0: Conversion is not complete 1: Conversion complete
[1]	RC_W0	**EOC:** End of conversion This bit is set by hardware at the end of a group channel conversion (regular or injected). It is cleared by software or by reading the ADC_DR. 0: Conversion is not complete 1: Conversion complete
[0]	RC_W0	**AWD:** Analog watchdog flag This bit is set by hardware when the converted voltage crosses the values programmed in the ADC_LTR and ADC_HTR registers. It is cleared by software. 0: No Analog watchdog event occurred 1: Analog watchdog event occurred

11.11.2 ADC control register 1(ADC_CR1)

Register	Address offset	Access	Reset value	Description
ADC_CR1	0x04	RW	0x0000_0000	ADC control register 1

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
AWDEN	JAWDEN	Reserved		DUALMOD[3:0]			
15	14	13	12	11	10	9	8
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN
7	6	5	4	3	2	1	0
JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				

ADC control register 1(ADC_CR1)bit description

Bit	Access	Description
[31:24]	R	Reserved, must be kept at reset value.
[23]	RW	**AWDEN:** Analog watchdog enable on regular channels This bit is set/reset by software. 0: Analog watchdog disabled on regular channels 1: Analog watchdog enabled on regular channels
[22]	RW	**JAWDEN:** Analog watchdog enable on injected channels This bit is set/reset by software. 0: Analog watchdog disabled on injected channels 1: Analog watchdog enabled on injected channels
[21:20]	R	Reserved, must be kept at reset value.
[19:16]	RW	**DUALMOD[3:0]:** Dual mode selection These bits are written by software to select the operating mode, 0000: Independent mode. 0001: Combined regular simultaneous + injected simultaneous mode 0010: Combined regular simultaneous + alternate trigger mode 0011: Combined injected simultaneous + fast interleaved mode 0100: Combined injected simultaneous + slow Interleaved mode 0101: Injected simultaneous mode only 0110: Regular simultaneous mode only 0111: Fast interleaved mode only 1000: Slow interleaved mode only 1001: Alternate trigger mode only **Note:** *These bits are reserved in ADC2 and ADC3.* *In dual mode, a change of channel configuration generates a restart that can produce a loss of synchronization. It is recommended to disable dual mode before any configuration change.*
[15:13]	RW	**DISCNUM[2:0]:** Discontinuous mode channel count These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger. 000 ~ 111: 1 ~ 8 channels
[12]	RW	**JAWDEN:** Discontinuous mode on injected channels This bit set and cleared by software to enable/disable discontinuous mode on injected group channels 0: Discontinuous mode on injected channels disabled 1: Discontinuous mode on injected channels enabled

[11]	RW	**DISCEN:** Discontinuous mode on regular channels This bit set and cleared by software to enable/disable Discontinuous mode on regular channels. 0: Discontinuous mode on regular channels disabled 1: Discontinuous mode on regular channels enabled
[10]	RW	**JAUTO:** Automatic Injected Group conversion This bit set and cleared by software to enable/disable automatic injected group conversion after regular group conversion 0: disabled 1: enabled
[9]	RW	**AWDSGL:** Enable the watchdog on a single channel in scan mode This bit set and cleared by software to enable/disable the analog watchdog on the channel identified by the AWDCH[4:0] bits. 0: Analog watchdog enabled on all channels 1: Analog watchdog enabled on a single channel
[8]	RW	**SCAN:** Scan mode This bit is set and cleared by software to enable/disable Scan mode. In Scan mode, the inputs selected through the ADC_SQRx or ADC_JSQRx registers are converted. 0: Scan mode disabled 1: Scan mode enabled **Note:** *An EOC or JEOC interrupt is generated only on the end of conversion of the last channel if the corresponding EOCIE or JEOCIE bit is set*
[7]	RW	**JEOCIE:** Interrupt enable for injected channels This bit is set and cleared by software to enable/disable the end of conversion interrupt for injected channels. 0: JEOC interrupt disabled 1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.
[6]	RW	**AWDIE:** Analog watchdog interrupt enable This bit is set and cleared by software to enable/disable the analog watchdog interrupt. 0: Analog watchdog interrupt disabled 1: Analog watchdog interrupt enabled
[5]	RW	**EOCIE:** Interrupt enable for EOC This bit is set and cleared by software 0: EOC interrupt disabled 1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.
[4:0]	RW	**AWDCH[4:0]:** Analog watchdog channel select bits These bits are set and cleared by software. 用于选择模拟看门狗保护的输入通道。 00000 ~ 10001: ADC analog Channel 0 ~ 17 Other values reserved. **Note:** *ADC1 analog Channel16 and Channel17 are internally connected to the temperature sensor and to* V_{REFINT}*, respectively.* *ADC2 analog inputs Channel16 and Channel17 are internally connected to* V_{SS} *,* *ADC3 analog inputs Channel9, Channel14, Channel15, Channel16 and Channel17 are connected to* V_{SS}

11.11.3 ADC control register 2 (ADC_CR2)

Register	Address offset	Access	Reset value	Description
ADC_CR2	0x08	RW	0x0000_0000	ADC control register 2

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2:0]			Reserved
15	14	13	12	11	10	9	8
JEXTTRIG	JEXTSEL[2::0]			ALIGN	Reserved		Reserved
7	6	5	4	3	2	1	0
Reserved						CONT	ADON

ADC control register 2 (ADC_CR2)bit description

Bit	Access	Description
[31:24]	R	Reserved, must be kept at reset value.
[23]	RW	**TSVREFE:** Temperature sensor and V_{REFINT} enable This bit is set and cleared by software, Only ADC1 is available. 0: Temperature sensor and V_{REFINT} channel disabled 1: Temperature sensor and V_{REFINT} channel enabled
[22]	RW	**SWSTART:** Start conversion of regular channels This bit is set by software to start conversion and cleared by hardware as soon as conversion starts.It starts a conversion of a group of regular channels if SWSTART is selected as trigger event by the EXTSEL[2:0] bits. 0: Reset state 1: Starts conversion of regular channels
[21]	RW	**JSWSTART:** Start conversion of injected channels This bit is set by software and cleared by software or by hardware as soon as the conversion starts. It starts a conversion of a group of injected channels (if JSWSTART is selected as trigger event by the JEXTSEL[2:0] bits.) 0: Reset state 1: Starts conversion of injected channels
[20]	RW	**EXTTRIG:** External trigger conversion mode for regular channels This bit is set and cleared by software to enable/disable the external trigger used to start conversion of a regular channel group. 0: Conversion on external event disabled 1: Conversion on external event enabled

[19:17]	RW	**EXTSEL[2:0]:** External event select for regular group These bits select the external event used to trigger the start of conversion of a regular group: For ADC1 and ADC2, the assigned triggers are: 000: Timer 1 CC1 event 001: Timer 1 CC2 event 010: Timer 1 CC3 event 011: Timer 2 CC2 event 100: Timer 3 TRGO event 101: Timer 4 CC4 event 110: EXTI line 11/TIM8_TRGO event 111: SWSTART For ADC3, the assigned triggers are: 000: Timer 3 CC1 event 001: Timer 2 CC3 event 010: Timer 1 CC3 event 011: Timer 8 CC1 event 100: Timer 8 TRGO event 101: Timer 5 CC1 event 110: Timer 5 CC3 event 111: SWSTART
[16]	R	Reserved, must be kept at reset value.
[15]	RW	**JEXTTRIG:** External trigger conversion mode for injectedchannels This bit is set and cleared by software to enable/disable the external trigger used to start conversion of an injected channel group 0: Conversion on external event disabled 1: Conversion on external event enabled

[14:12]	RW	**JEXTSEL[2:0]:** External event select for injected group These bits select the external event used to trigger the start of conversion of an injected group: For ADC1 and ADC2 the assigned triggers are: 000: Timer 1 TRGO event 001: Timer 1 CC4 event 010: Timer 2 TRGO event 011: Timer 2 CC1 event 100: Timer 3 CC4 event 101: Timer 4 TRGO event 110: EXTI line15/TIM8_CC4 event 111: JSWSTART For ADC3 the assigned triggers are: 000: Timer 1 TRGO event 001: Timer 1 CC4 event 010: Timer 4 CC3 event 011: Timer 8 CC2 event 100: Timer 8 CC4 event 101: Timer 5 TRGO event 110: Timer 5 CC4 event 111: JSWSTART
[11]	RW	**ALIGN:** Data alignment This bit is set and cleared by software. 0: Right Alignment 1: Left Alignment
[10:9]	R	Reserved, must be kept at reset value.
[8]	RW	**DMA:** Direct memory access mode enable This bit is set and cleared by software， DMA controller。 0: DMA mode disabled 1: DMA mode enabled **Note：** *Only ADC1 and ADC3 can generate a DMA request.*
[7:2]	R	Reserved, must be kept at reset value.
[1]	RW	**CONT:** Continuous conversion This bit is set and cleared by software.After the bit set 1, the conversion continues until the bit is set to 0. 0: Single conversion mode 1: Continuous conversion mode
[0]	RW	**ADON:** A/D converter ON/OFF This bit is set and cleared by software. If this bit holds a value of zero and a 1 is written to it then it wakes up the ADC from Power Down state.Conversion starts when this bit holds a value of 1 and a 1 is written to it. The application should allow a delay of t_{STAB} between power up and start of conversion.Refer to 11.4-1 0: Disable ADC and go to power down mode 1: Enable ADC and to start conversion **Note：** *If any other bit in this register apart from ADON is changed at the same time, then conversion is not triggered. This is to prevent triggering an erroneous conversion.*

11.11.4 ADC sample time register 1(ADC_SMPR1)

Register	Address offset	Access	Reset value	Description
ADC_SMPR1	0x0C	RW	0x0000_0000	ADC sample time register 1

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
15	14	13	12	11	10	9	8
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2]
7	6	5	4	3	2	1	0
SMP12[1:0]		SMP11[2:0]			SMP10[2:0]		

ADC sample time register 1(ADC_SMPR1)bit description

Bit	Access	Description
[31:24]	R	Reserved, must be kept at reset 0
[23:0]	RW	**SMPx[2:0]:** Channel x Sample time selection, These bits are written by software to select the sample time individually for each channel. During sample cycles channel selection bits must remain unchanged. 000: 4.5 cycles 001: 7.5 cycles 010: 13.5 cycles 011: 28.5 cycles 100: 41.5 cycles 101: 55.5 cycles 110: 71.5 cycles 111: 239.5 cycles **Note:** *ADC1 analog Channel16 and Channel 17 are internally connected to the temperature sensor and to VREFINT , respectively.* *ADC2 analog input Channel16 and Channel17 are internally connected to VSS.* *ADC3 analog inputs Channel14, Channel15, Channel16 and Channel17 are connected to VSS*

11.11.5 ADC sample time register 2(ADC_SMPR2)

Register	Address offset	Access	Reset value	Description
ADC_SMPR2	0x10	RW	0x0000_0000	ADC sample time register 2

31	30	29	28	27	26	25	24
Reserved		SMP9[2:0]			SMP8[2:0]		
23	22	21	20	19	18	17	16
SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
15	14	13	12	11	10	9	8
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2]
7	6	5	4	3	2	1	0
SMP2[1:0]		SMP1[2:0]			SMP0[2:0]		

ADC sample time register 2(ADC_SMPR2)bit description

Bit	Access	Description

[31:30]	R	Reserved, must be kept at reset 0
[29:0]	RW	**SMPx[2:0]:** Channel x Sample time selection, These bits are written by software to select the sample time individually for each channel. During sample cycles channel selection bits must remain unchanged. 000: 4.5 cycles 001: 7.5 cycles 010: 13.5 cycles 011: 28.5 cycles 100: 41.5 cycles 101: 55.5 cycles 110: 71.5 cycles 111: 239.5 cycles **Note:** *ADC3 analog input Channel9 is connected to VSS*

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

Register	Address offset	Access	Reset value	Description
ADC_JOFRx (x = 1,2,3,4)	0x10 + x*4	RW	0x0000_0000	ADC injected channel data offset register x

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

ADC injected channel data offset register x(ADC_JOFRx)bit description

Bit	Access	Description
[31:12]	R	Reserved, must be kept at reset 0
[11:0]	RW	**JOFFSETx[11:0]:** Data offset for injected channel x These bits are written by software to define the offset to be subtracted from the raw converted data when converting injected channels. The conversion result can be read from in the ADC_JDRx registers.

11.11.7 ADC watchdog high threshold register (ADC_HTR)

Register	Address offset	Access	Reset value	Description
ADC_HTR	0x24	RW	0x0000_0000	ADC watchdog high threshold register

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

ADC watchdog high threshold register(ADC_HTR)bit description

Bit	Access	Description
[31:12]	R	Reserved, must be kept at reset 0
[11:0]	RW	**HT[11:0]:** Analog watchdog high threshold These bits are written by software to define the high threshold for the analog watchdog **Note:** *The software can write to these registers when an ADC conversion is ongoing. The programmed value will be effective when the next conversion is complete. Writing to this register is performed with a write delay that can create uncertainty on the effective time at which the new value is programmed.*

11.11.8 ADC watchdog low threshold register (ADC_LTR)

Register	Address offset	Access	Reset value	Description
ADC_LTR	0x28	RW	0x0000_0000	ADC watchdog low threshold register

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

ADC watchdog low threshold register(ADC_LTR)bit description

Bit	Access	Description
[31:12]	R	Reserved, must be kept at reset 0
[11:0]	RW	**LT[11:0]:** Analog watchdog low threshold These bits are written by software to define the low threshold for the analog watchdog. **Note:** *The software can write to these registers when an ADC conversion is ongoing. The programmed value will be effective when the next conversion is complete. Writing to this register is performed with a write delay that can create uncertainty on the effective time at which the new value is programmed.*

11.11.9 ADC regular sequence register 1 (ADC_SQR1)

Register	Address offset	Access	Reset value	Description
ADC_SQR1	0x2C	RW	0x0000_0000	ADC regular sequence register 1

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
L[3:0]				SQ16[4:1]			
15	14	13	12	11	10	9	8
SQ16[0]	SQ15[4:0]					SQ14[4:3]	
7	6	5	4	3	2	1	0
SQ14[2:0]			SQ13[4:0]				

ADC regular sequence register 1 (ADC_SQR1)bit description

Bit	Access	Description
[31:24]	R	Reserved, must be kept at reset 0
[23:20]	RW	**L[3:0]:** Regular channel sequence length These bits are written by software to define the total number of conversions in the regular channel conversion sequence 0000 ~ 1111: 1 ~ 16 conversions
[19:15]	RW	**SQ16[4:0]:** 16th conversion in regular sequence These bits are written by software with the channel number (0~17) assigned as the 16th in the conversion sequence.
[14:10]	RW	**SQ15[4:0]:** 15th conversion in regular sequence
[9:5]	RW	**SQ14[4:0]:** 14th conversion in regular sequence
[4:0]	RW	**SQ13[4:0]:** 13th conversion in regular sequence

11.11.10 ADC regular sequence register 2 (ADC_SQR2)

Register	Address offset	Access	Reset value	Description
ADC_SQR2	0x30	RW	0x0000_0000	ADC regular sequence register 2

31	30	29	28	27	26	25	24
Reserved		SQ12[4:0]					SQ11[4]
23	22	21	20	19	18	17	16
SQ11[3:0]				SQ10[4:1]			
15	14	13	12	11	10	9	8
SQ10[0]	SQ9[4:0]					SQ8[4:2]	
7	6	5	4	3	2	1	0
SQ8[2:0]			SQ7[4:0]				

ADC regular sequence register 2 (ADC_SQR2)bit description

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

11.11.11 ADC regular sequence register 3 (ADC_SQR3)

Register	Address offset	Access	Reset value	Description
ADC_SQR3	0x34	RW	0x0000_0000	ADC regular sequence register 3

31	30	29	28	27	26	25	24
Reserved		SQ6[4:0]					SQ5[4]
23	22	21	20	19	18	17	16
SQ5[3:0]				SQ4[4:1]			
15	14	13	12	11	10	9	8
SQ4[0]	SQ3[4:0]					SQ2[4:2]	
7	6	5	4	3	2	1	0
SQ2[2:0]			SQ1[4:0]				

ADC regular sequence register 3(ADC_SQR3)bit description

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

11.11.12 ADC injected sequence register(ADC_JSQR)

Register	Address offset	Access	Reset value	Description
ADC_JSQR	0x38	RW	0x0000_0000	ADC injected sequence register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		JL[1:0]		JSQ4[4:1]			
15	14	13	12	11	10	9	8
JSQ4[0]	JSQ3[4:0]					JSQ2[4:2]	
7	6	5	4	3	2	1	0
JSQ2[2:0]			JSQ1[4:0]				

ADC injected sequence register (ADC_JSQR)bit description

Bit	Access	Description
[31:22]	R	Reserved, must be kept at reset 0
[21:20]	RW	**JL[1:0]:** Injected sequence length These bits are written by software to define the total number of conversions in the injected channel conversion sequence. 00 : 1 conversion 01 : 2 conversions 10 : 3 conversions 11 : 4 conversions

[19:15]	RW	**JSQ4[4:0]:** 4th conversion in injected sequence(JL[1:0]=11) These bits are written by software with the channel number (0..17) assigned as the fourth in the sequence to be converted.
[14:10]	RW	**JSQ3[4:0]:** 3rd conversion in injected sequence(JL[1:0]=11)
[9:5]	RW	**JSQ2[4:0]:** 2nd conversion in injected sequence(JL[1:0]=11)
[4:0]	RW	**JSQ1[4:0]:** 1st conversion in injected sequence(JL[1:0]=11)

Note: *Unlike a regular conversion sequence, if JL[1:0] length is less than four, the channels are converted in a sequence starting from (4-JL).*:

- *When JL = 3 (4 injection conversions in the sequencer), the ADC converts the channel in the following order*: *JSQ1 [4: 0] » JSQ2 [4: 0] » JSQ3 [4: 0] » JSQ4 [4: 0]*
- *When JL = 2 (3 injection conversions in the sequencer), the ADC converts the channel in the following order*: *JSQ2 [4: 0] » JSQ3 [4: 0] » JSQ4 [4: 0]*
- *When JL = 1 (2 injection conversions in the sequencer), the ADC converts the channel in the following order*: *JSQ3 [4: 0] » JSQ4 [4: 0]*
- *When JL = 0 (1 injection conversions in the sequencer), the ADC converts the channel in the following order*: *Example: ADC_JSQR[21:0] = 10 00011 00011 00111 00010 means that a scan conversion will convert the following channel sequence: 7, 3, 3. (not 2, 7, 3)*

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

Register	Address offset	Access	Reset value	Description
ADC_JDRx (x = 1,2,3,4)	0x38 + x*4	R	0x0000_0000	ADC injected data register x

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

ADC injected data register x(ADC_JDRx)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset 0
[15:0]	R	**JDATA[15:0]:** Injected data These bits are read only. They contain the conversion result from injected channel x. The data is left or right-aligned as shown in 11.5-2 and 11.5-1

11.11.14 ADC regular data register(ADC_DR)

Register	Address offset	Access	Reset value	Description
ADC_DR	0x4C	R	0x0000_0000	ADC regular data register

31	30	29	28	27	26	25	24
ADC2DATA[15:8]							
23	22	21	20	19	18	17	16
ADC2DATA[7:0]							
15	14	13	12	11	10	9	8
DATA[15:8]							
7	6	5	4	3	2	1	0
DATA[7:0]							

ADC regular data register(ADC_DR)bit description

Bit	Access	Description
[31:16]	R	**ADC2DATA[15:0]:** ADC2 data In ADC1: In dual mode, these bits contain the regular data of ADC2. Refer to Section 11.9 In ADC2 and ADC3: these bits are not used.
[15:0]	R	**DATA[15:0]:** Regular data These bits are read only. They contain the conversion result from the regular channels.

12 Digital-to-analog converter (DAC)

12.1 DAC introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC has two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operation. An input reference pin VREF+ (shared with ADC) is available for better resolution.

12.2 DAC main features

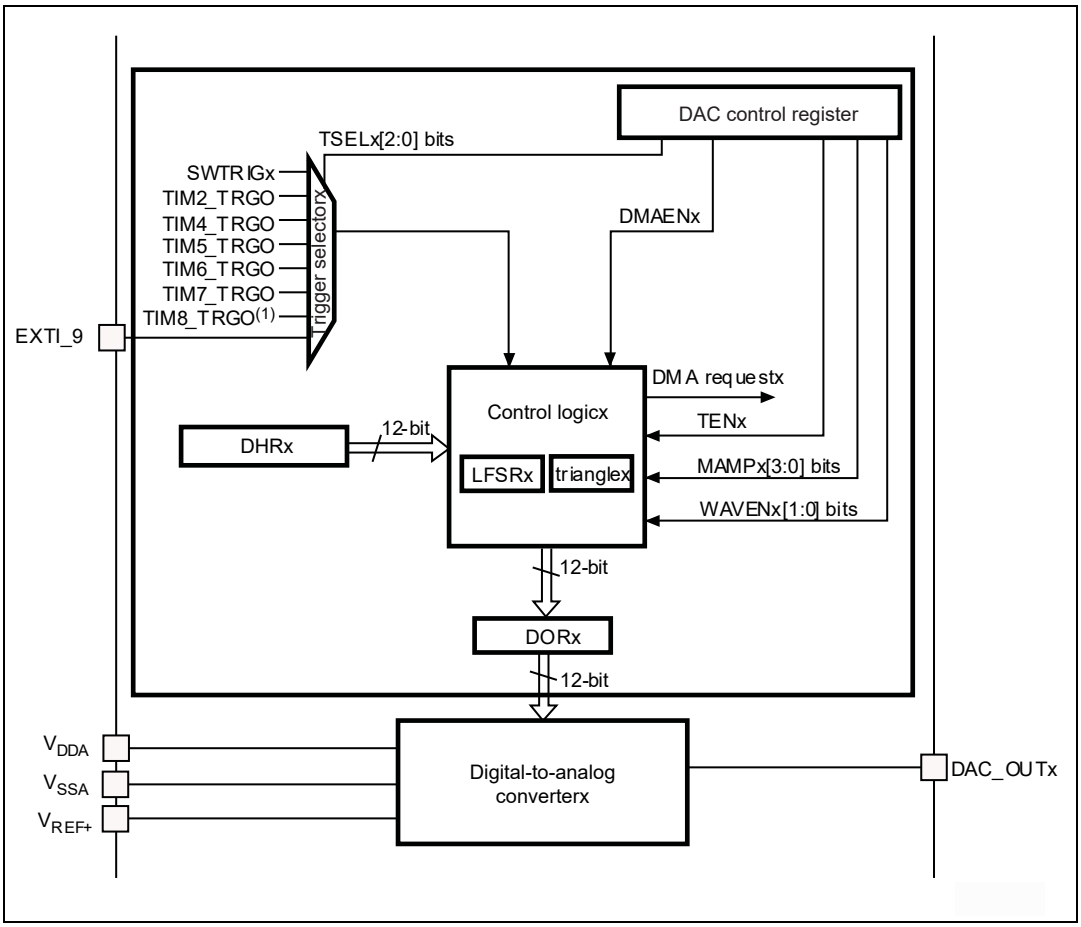
- Two DAC converters: one output channel each
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave generation
- Triangular-wave generation
- Dual DAC channel independent or simultaneous conversions
- DMA capability for each channel
- External triggers for conversion
- Input voltage reference VREF+

Tab 12.2-1 DAC pins

Name	Signal type	Remarks
VREF+	Input, analog reference positive	higher/positive reference voltage for DAC,2.4V $\leqslant V_{REF+} \leqslant V_{DDA}$(3.3V)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

Note: *Once DAC channelx is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the analog converter output (DAC_OUTx). To avoid parasitic consumption, the PA4 or PA5 pin should first be configured to analog (AIN).*

Fig 12.2-1 DAC channel block diagram



Note: *In connectivity line devices, the TIM8_TRGO trigger is replaced by TIM3_TRGO*

12.3 DAC functional description

12.3.1 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a startup time tWAKEUP.

Note: *The ENx bit enables the analog DAC Channelx macrocell only. The DAC Channelx digital interface is enabled even if the ENx bit is reset.*

12.3.2 DAC output buffer enable

The DAC integrates two output buffers that can be used to reduce the output impedance, and to drive external loads directly without having to add an external operational amplifier. Each DAC channel output buffer can be enabled and disabled using the corresponding BOFFx bit in the DAC_CR register.

12.3.3 DAC data format

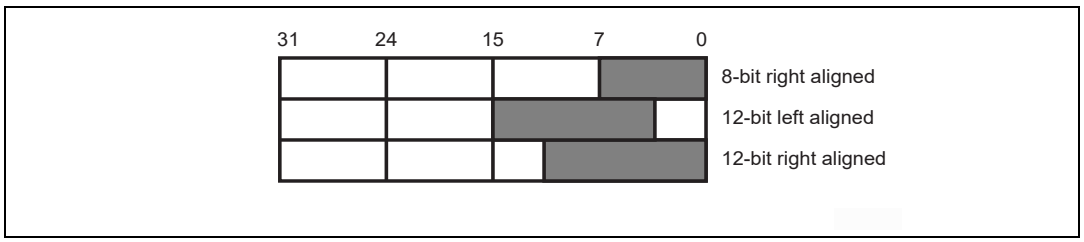
Depending on the selected configuration mode, the data has to be written in the specified register as described below:

Single DAC channelx, there are three possibilities:

- 8-bit right alignment: user has to load data into DAC_DHR8Rx [7:0] bits (stored into DHRx[11:4] bits)
- 12-bit left alignment: user has to load data into DAC_DHR12Lx [15:4] bits (stored into DHRx[11:0] bits)
- 12-bit right alignment: user has to load data into DAC_DHR12Rx [11:0] bits (stored into DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user will be shifted and stored into the DHRx (Data Holding Registerx, that are internal non-memory-mapped registers). The DHRx register will then be loaded into the DORx register either automatically, by software trigger or by an external event trigger.

Fig 12.3-1 Data registers in single DAC channel mode

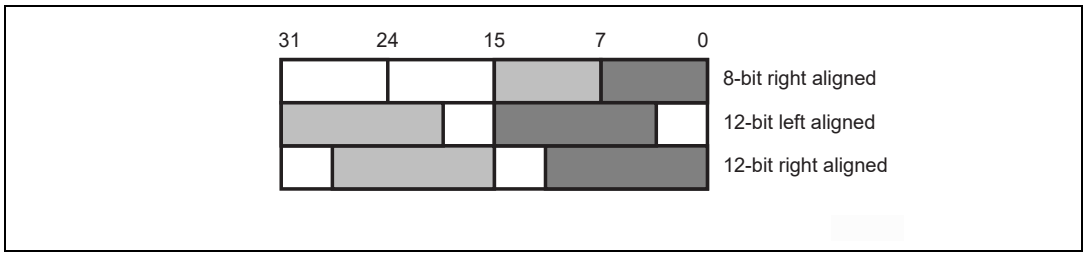


Dual DAC channels, there are three possibilities:

- 8-bit right alignment: data for DAC channel1 to be loaded into DAC_DHR8RD [7:0] bits (stored into DHR1[11:4] bits) and data for DAC channel2 to be loaded into DAC_DHR8RD [15:8] bits (stored into DHR2[11:4] bits)）
- 12-bit left alignment: data for DAC channel1 to be loaded into DAC_DHR12LD [15:4] bits (stored into DHR1[11:0] bits) and data for DAC channel2 to be loaded into DAC_DHR12LD [31:20] bits (stored into DHR2[11:0] bits)
- 12-bit right alignment: data for DAC channel1 to be loaded into DAC_DHR12RD [11:0] bits (stored into DHR1[11:0] bits) and data for DAC channel2 to be loaded into DAC_DHR12RD [27:16] bits (stored into DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user will be shifted and stored into the DHR1 and DHR2（Data Holding Registers, that are internal non- memory-mapped registers）. The DHR1 and DHR2 registers will then be loaded into the DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

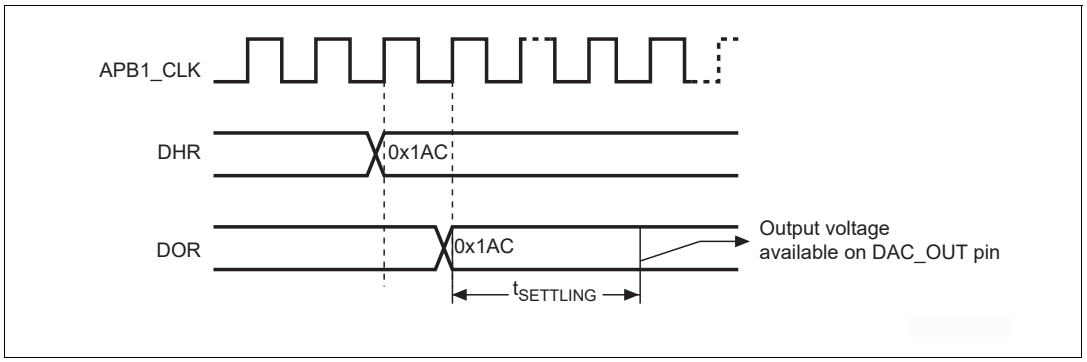
Fig 12.3-2 Data registers in dual DAC channel mode



12.3.4 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write on DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12LD or DAC_DHR12RD).
Data stored into the DAC_DHRx register are automatically transferred to the DAC_DORx register after one APB1 clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three APB1 clock cycles later.
When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time of $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

Fig 12.3-3 Timing diagram for conversion with trigger disabled



12.3.5 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}. The analog output voltages on each DAC channel pin are determined by the equation:

$$V_{OUT} = \frac{V_{REF} * DOR}{4096} \tag{12.1}$$

12.3.6 DAC trigger selection

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[2:0] control bits determine which one, out of 8 possible events, will trigger conversion。

Tab 12.3-1 DAC External triggers

Source	Type	TSEL[2:0]
Timer 6 TRGO event	Internal signal from on-chip timers	000
Timer 3 TRGO event		001
Timer 7 TRGO event		010
Timer 5 TRGO event		011
Timer 2 TRGO event		100
Timer 4 TRGO event		101
EXTI line9	External pin	110
SWTRIG（Software）	Software control bit	111

Each time a DAC interface detects a rising edge on the selected timer TRGO output, or on the selected external interrupt line 9, the last data stored into the DAC_DHRx register is transferred into the DAC_DORx register. The DAC_DORx register is updated three APB1 cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: *TSELx[2:0] bit cannot be changed when the ENx bit is set. When software trigger is selected, it takes only one APB1 clock cycle for DAC_DHRx-to-DAC_DORx register transfer*

12.3.7 DMA request

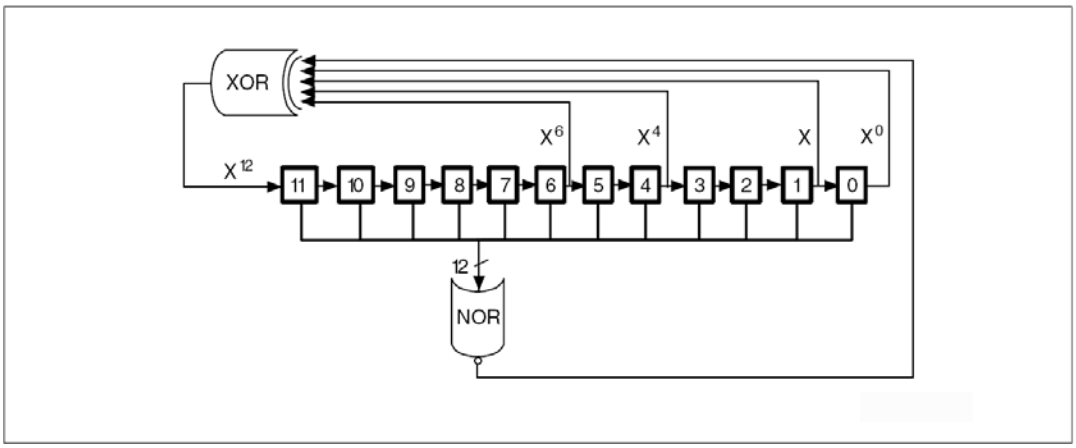
Each DAC channel has a DMA capability.
A DAC DMA request is generated when an external trigger (but not a software trigger) occurs while the DMAENx bit is set. The value of the DAC_DHRx register is then transferred to the DAC_DORx register.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, you should set only the corresponding DMAENx bit. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

12.3.8 Noise generation

In order to generate a variable-amplitude pseudonoise, a Linear Feedback Shift Register is available. The DAC noise generation is selected by setting WAVEx[1:0] to “01”. The preloaded value in the LFSR is 0xAAA. This register is updated, three APB1 clock cycles after each trigger event, following a specific calculation algorithm.

Fig 12.3-4 DAC LFSR register calculation algorithm

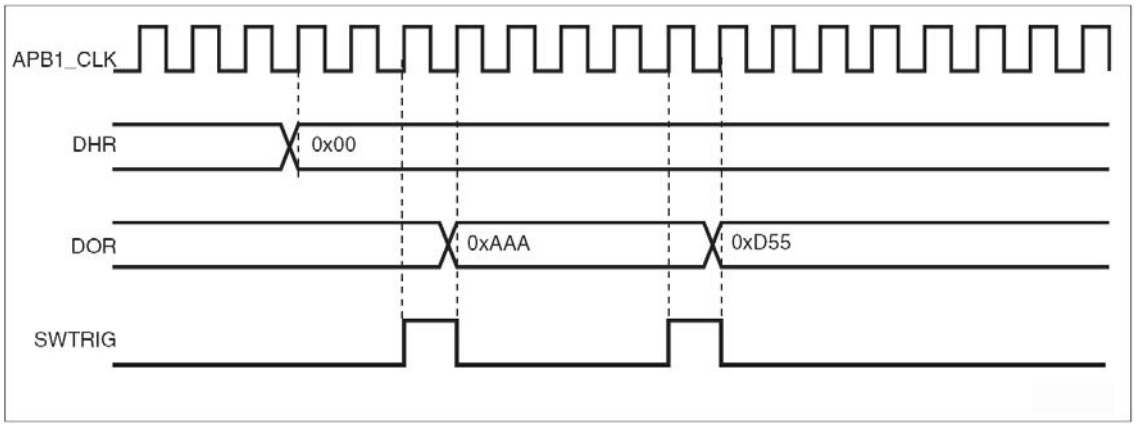


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then stored into the DAC_DORx register.

If LFSR is 0x0000, a ‘1’ is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Fig 12.3-5 DAC conversion (SW trigger enabled) with LFSR wave generation



Note: *DAC trigger must be enabled for noise generation, by setting the TENx bit in the DAC_CR register.*

12.3.9 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to “10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three APB1 clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is stored into the DAC_DORx register. The triangle counter is incremented while it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and

so on.

It is possible to reset triangle wave generation by resetting WAVEx[1:0] bits.

Fig 12.3-6 DAC triangle wave generation

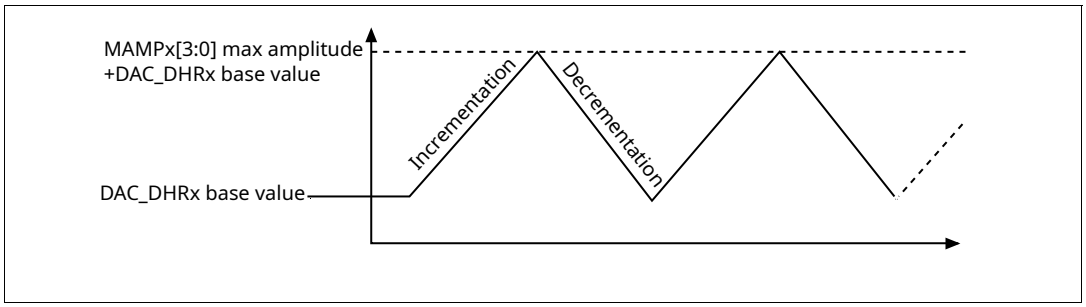
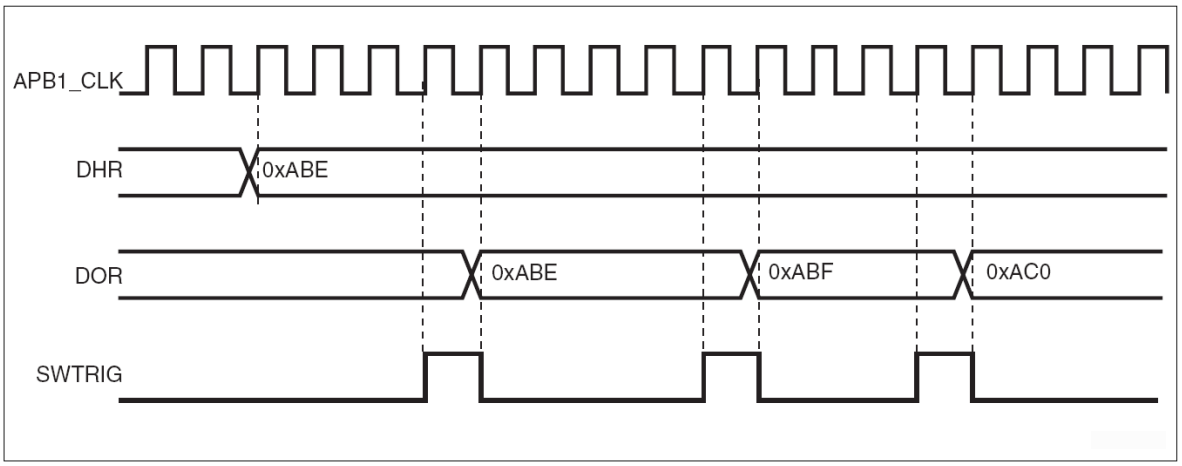


Fig 12.3-7 DAC conversion (SW trigger enabled) with triangle wave generation



Note: *DAC trigger must be enabled for noise generation, by setting the TENx bit in the DAC_CR register.MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.*

12.4 Dual DAC channel conversion

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time.

Eleven possible conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

12.4.1 Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- The two DAC channels TSELx[2:0] bits are set to different values to select different trigger sources
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or

DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three APB1 clock cycles later).
When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three APB1 clock cycles later).

12.4.2 Independent trigger with same LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- The two DAC channels TSELx[2:0] bits are set to different values to select different trigger sources
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.
When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated

12.4.3 Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- The two DAC channels TSELx[2:0] bits are set to different values to select different trigger sources
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR masks values in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.
When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

12.4.4 Independent trigger with same triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- The two DAC channels TSELx[2:0] bits are set to different values to select different trigger sources

- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same maximum amplitude value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)。

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.
When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

12.4.5 Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- The two DAC channels TSELx[2:0] bits are set to different values to select different trigger sources
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different maximum amplitude values in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.
When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register part and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

12.4.6 Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

In this configuration, one APB1 clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

12.4.7 Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSELx[2:0] bits

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three APB1 clock cycles).

12.4.8 Simultaneous trigger with same LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Set the TSELx[2:0] bit of both DAC channels to the same value to select the same trigger source
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

12.4.9 Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Set the TSELx[2:0] bit of both DAC channels to the same value to select the same trigger source
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR masks values using the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When the trigger signal arrives, the LFSR1 and LFSR2 counters with the same mask are added to the DHR1 and DHR2 registers respectively, and the result is sent to the DAC_DOR1 and DAC_DOR2 respectively after three APB1 clock cycles. Then the LFSR1 and LFSR2 counters are updated.

12.4.10 Simultaneous trigger with same triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Set the TSELx[2:0] bit of both DAC channels to the same value to select the same trigger source
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and the same maximum amplitude value using the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The

DAC channel1 triangle counter is then updated. At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

12.4.11 Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Set the TSELx[2:0] bit of both DAC channels to the same value to select the same trigger source
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When the trigger signal arrives, the trigonometric counter values of DAC channels 1 and 2 configured with trigonometric amplitude by MAMP1[3:0] and MAMP2[3:0] are added to the DHR1 and DHR2 registers, respectively. After three APB1 clock cycles, the results are sent to DAC_DOR1 and DAC_DOR2 respectively. Subsequently the DAC channel 1 and DAC channel 2 triangle counters are updated

12.5 DAC Register

Tab 12.5-1 DAC register map

Offset	Register	Reset value	Description
DAC: DAC_BA = 0x4000_7400			
0x00	DAC_CR	0x0000_0000	DAC control register
0x04	DAC_SWTRIGR	0x0000_0000	DAC software trigger register
0x08	DAC_DHR12R1	0x0000_0000	DAC channel1 12-bit right-aligned data holding register
0x0C	DAC_DHR12L1	0x0000_0000	DAC channel1 12-bit left aligned data holding register
0x10	DAC_DHR8R1	0x0000_0000	DAC channel1 8-bit right aligned data holding register
0x14	DAC_DHR12R2	0x0000_0000	DAC channel2 12-bit right aligned data holding register
0x18	DAC_DHR12L2	0x0000_0000	DAC channel2 12-bit left aligned data holding register
0x1C	DAC_DHR8R2	0x0000_0000	DAC channel2 8-bit right-aligned data holding register
0x20	DAC_DHR12RD	0x0000_0000	Dual DAC 12-bit right-aligned data holding register
0x24	DAC_DHR12LD	0x0000_0000	DUAL DAC 12-bit left aligned data holding register
0x28	DAC_DHR8RD	0x0000_0000	DUAL DAC 8-bit right aligned data holding register
0x2C	DAC_DOR1	0x0000_0000	DAC channel1 data output register
0x30	DAC_DOR2	0x0000_0000	DAC channel2 data output register

12.5.1 DAC control register(DAC_CR)

Register	Address offset	Access	Reset value	Description
DAC_CR	0x00	RW	0x0000_0000	DAC control register

31	30	29	28	27	26	25	24
Reserved			DMAEN2	MAMP2[3:0]			
23	22	21	20	19	18	17	16
WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
15	14	13	12	11	10	9	8
Reserved			DMAEN1	MAMP1[3:0]			
7	6	5	4	3	2	1	0
WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1

DAC control register(DAC_CR)bit description

Bit	Access	Description
[31:29]	R	Reserved, must be held at reset value.
[28]	RW	**DMAEN2:** DAC channel2 DMA enable This bit is set and cleared by software 0: DAC channel2 DMA mode disabled 1: DAC channel2 DMA mode enabled
[27:24]	RW	**MAMP2[3:0]:** DAC channel2 mask/amplitude selector These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode. 0000: Unmask bit0 of LFSR/ Triangle Amplitude equal to 1 0001: Unmask bits[1:0] of LFSR/ Triangle Amplitude equal to 3 0010: Unmask bits[2:0] of LFSR/ Triangle Amplitude equal to 7 0011: Unmask bits[3:0] of LFSR/ Triangle Amplitude equal to 15 0100: Unmask bits[4:0] of LFSR/ Triangle Amplitude equal to 31 0101: Unmask bits[5:0] of LFSR/ Triangle Amplitude equal to 63 0110: Unmask bits[6:0] of LFSR/ Triangle Amplitude equal to 127 0111: Unmask bits[7:0] of LFSR/ Triangle Amplitude equal to 255 1000: Unmask bits[8:0] of LFSR/ Triangle Amplitude equal to 511 1001: Unmask bits[9:0] of LFSR/ Triangle Amplitude equal to 1023 1010: Unmask bits[10:0] of LFSR/ Triangle Amplitude equal to 2047 ⩾1011: Unmask bits[11:0] of LFSR/ Triangle Amplitude equal to 4095
[23:22]	RW	**WAVE2[1:0]:** DAC channel2 noise/triangle wave generationenable These bits are set/reset by software. 00: wave generation disabled 01: Noise wave generation enabled 1x: Triangle wave generation enabled **Note:** *only used if bit TEN2 = 1 (DAC channel2 trigger enabled)*

[21:19]	RW	**TSEL2[2:0]:** DAC channel2 trigger selection These bits select the external event used to trigger DAC channel2 000: Timer 6 TRGO event 001: Timer 8 TRGO event 010: Timer 7 TRGO event 011: Timer 5 TRGO event 100: Timer 2 TRGO event 101: Timer 4 TRGO event 110: External line9 111: Software trigger **Note:** *only used if bit TEN2 = 1 (DAC channel2 trigger enabled)*
[18]	RW	**TEN2:** DAC channel2 trigger enable This bit set and cleared by software to enable/disable DAC channel2 trigger 0: DAC channel2 trigger disabled and data written into DAC_DHRx register is transferred one APB1 clock cycle later to the DAC_DOR2 register. 1: DAC channel2 trigger enabled and data transfer from DAC_DHRx register is transferred three APB1 clock cycles later to the DAC_DOR2 register. **Note:** *When software trigger is selected, it takes only one APB1 clock cycle for DAC_DHRx to DAC_DOR2 register transfer.*
[17]	RW	**BOFF2:** DAC channel2 output buffer disable This bit set and cleared by software to enable/disable DAC channel2 output buffer. 0: DAC channel2 output buffer enabled 1: DAC channel2 output buffer disabled
[16]	RW	**EN2:** DAC channel2 enable This bit set and cleared by software to enable/disable DAC channel2. 0: DAC channel2 disabled 1: DAC channel2 enabled
[15:13]	R	Reserved, must be held at reset value.
[12]	RW	**DMAEN1:** DAC channel1 DMA enable This bit is set and cleared by software. 0: DAC channel1 DMA mode disabled 1: DAC channel1 DMA mode enabled

[11:8]	RW	**MAMP1[3:0]:** DAC channel1 mask/amplitude selector These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode. 0000: Unmask bit0 of LFSR/ Triangle Amplitude equal to 1 0001: Unmask bits[1:0] of LFSR/ Triangle Amplitude equal to 3 0010: Unmask bits[2:0] of LFSR/ Triangle Amplitude equal to 7 0011: Unmask bits[3:0] of LFSR/ Triangle Amplitude equal to 15 0100: Unmask bits[4:0] of LFSR/ Triangle Amplitude equal to 31 0101: Unmask bits[5:0] of LFSR/ Triangle Amplitude equal to 63 0110: Unmask bits[6:0] of LFSR/ Triangle Amplitude equal to 127 0111: Unmask bits[7:0] of LFSR/ Triangle Amplitude equal to 255 1000: Unmask bits[8:0] of LFSR/ Triangle Amplitude equal to 511 1001: Unmask bits[9:0] of LFSR/ Triangle Amplitude equal to 1023 1010: Unmask bits[10:0] of LFSR/ Triangle Amplitude equal to 2047 ⩾1011: Unmask bits[11:0] of LFSR/ Triangle Amplitude equal to 4095
[7:6]	RW	**WAVE1[1:0]:** DAC channel1 noise/triangle wave generation enable These bits are set/reset by software. 00: wave generation disabled 01: Noise wave generation enabled 1x: Triangle wave generation enabled **Note:** *only used if bit TEN1 = 1 (DAC channel1 trigger enabled)*
[5:3]	RW	**TSEL1[2:0]:** DAC channel1 trigger selection These bits select the external event used to trigger DAC channel1 000: Timer 6 TRGO event 001: Timer 8 TRGO event 010: Timer 7 TRGO event 011: Timer 5 TRGO event 100: Timer 2 TRGO event 101: Timer 4 TRGO event 110: External line9 111: Software trigger **Note:** *only used if bit TEN1 = 1 (DAC channel1 trigger enabled)*)
[2]	RW	**TEN1:** DAC channel1 trigger enable This bit set and cleared by software to enable/disable DAC channel1 trigger 0: DAC channel1 trigger disabled and data written into DAC_DHRx register is transferred one APB1 clock cycle later to the DAC_DOR1 register. 1: DAC channel1 trigger enabled and data transfer from DAC_DHRx register is transferred three APB1 clock cycles later to the DAC_DOR1 register **Note:** *When software trigger is selected, it takes only one APB1 clock cycle for DAC_DHRx to DAC_DOR1 register transfer*
[1]	RW	**BOFF1:** DAC channel1 output buffer disable This bit set and cleared by software to enable/disable DAC channel1 output buffer. 0: DAC channel1 output buffer enabled 1: DAC channel1 output buffer disabled

[0]	RW	**EN1:** DAC channel1 enable This bit set and cleared by software to enable/disable DAC channel1. 0: DAC channel1 disabled 1: DAC channel1 enabled

12.5.2 DAC software trigger register(DAC_SWTRIGR)

Register	Address offset	Access	Reset value	Description
DAC_SWTRIGR	0x04	RW	0x0000_0000	DAC software trigger Register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						SWTRIG2	SWTRIG1

DAC software trigger register(DAC_SWTRIGR)bit description

Bit	Access	Description
[31:2]	R	Reserved, must be held at reset value.
[1]	RW	**SWTRIG2:** DAC channel2 software trigger This bit is set and cleared by software to enable/disable the software trigger. 0: Software trigger disabled 1: Software trigger enabled **Note:** *This bit is reset by hardware (one APB1 clock cycle later) once the DAC_DHR2 register value is loaded to the DAC_DOR2 register.*
[0]	RW	**SWTRIG1:** DAC channel1 software trigger This bit is set and cleared by software to enable/disable the software trigger. 0: Software trigger disabled 1: Software trigger enabled **Note:** *This bit is reset by hardware (one APB1 clock cycle later) once the DAC_DHR1 register value is loaded to the DAC_DOR1 register.*

12.5.3 DAC channel1 12-bit right-aligned data holding register(DAC_DHR12R1)

Register	Address offset	Access	Reset value	Description
DAC_DHR12R1	0x08	RW	0x0000_0000	DAC channel1 12-bit right-aligned data holding register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				DACC1DHR[11:8]			
7	6	5	4	3	2	1	0
DACC1DHR[7:0]							

DAC channel1 12-bit right-aligned data holding register(DAC_DHR12R1)bit description

Bit	Access	Description
[31:12]	R	Reserved, must be held at reset value.
[11: 0]	RW	**DACC1DHR[11:0]:** DAC channel1 12-bit right-aligned data These bits are written by software which specify 12-bit data for DAC channel1.

12.5.4 DAC channel1 12-bit left aligned data holding register(DAC_DHR12L1)

Register	Address offset	Access	Reset value	Description
DAC_DHR12L1	0x0C	RW	0x0000_0000	DAC channel1 12-bit left aligned data holding register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
DACC1DHR[11:4]							
7	6	5	4	3	2	1	0
DACC1DHR[3:0]				Reserved			

DAC channel1 12-bit left aligned data holding register(DAC_DHR12L1)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15: 4]	RW	**DACC1DHR[11:0]:** DAC channel1 12-bit right-aligned data These bits are written by software which specify 12-bit data for DAC channel1.
[3:0]	R	Reserved

12.5.5 DAC channel1 8-bit right aligned data holding register(DAC_DHR8R1)

Register	Address offset	Access	Reset value	Description
DAC_DHR8R1	0x10	RW	0x0000_0000	DAC channel1 8-bit right aligned data holding register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
DACC1DHR[7:0]							

DAC channel1 8-bit right aligned data holding register(DAC_DHR8R1)bit description

Bit	Access	Description
[31:8]	R	Reserved
[7：0]	RW	**DACC1DHR[7:0]:** DAC channel1 8-bit right-aligned data These bits are written by software which specify 8-bit data for DAC channel1.

12.5.6 DAC channel2 12-bit right aligned data holding register(DAC_DHR12R2)

Register	Address offset	Access	Reset value	Description
DAC_DHR12R2	0x14	RW	0x0000_0000	DAC channel2 12-bit right aligned data holding register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				DACC2DHR[11:8]			
7	6	5	4	3	2	1	0
DACC2DHR[7:0]							

DAC channel2 12-bit right aligned data holding register(DAC_DHR12R2)bit description

Bit	Access	Description
[31:12]	R	Reserved
[11：0]	RW	**DACC2DHR[11:0]:** DAC channel2 12-bit right-aligned data These bits are written by software which specify 12-bit data for DAC channel2.

12.5.7 DAC channel2 12-bit left aligned data holding register(DAC_DHR12L2)

Register	Address offset	Access	Reset value	Description
DAC_DHR12L2	0x18	RW	0x0000_0000	DAC channel2 12-bit left aligned data holding register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
DACC2DHR[11:4]							
7	6	5	4	3	2	1	0
DACC2DHR[3:0]				Reserved			

DAC channel2 12-bit left aligned data holding register(DAC_DHR12L2)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15：4]	RW	**DACC2DHR[11:0]:** DAC channel2 12-bit left-aligned data These bits are written by software which specify 12-bit data for DAC channel2.
[3:0]	R	Reserved

12.5.8 DAC channel2 8-bit right-aligned data holding register(DAC_DHR8R2)

Register	Address offset	Access	Reset value	Description
DAC_DHR8R2	0x1C	RW	0x0000_0000	DAC channel2 8-bit right-aligned data holding register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
DACC2DHR[7:0]							

DAC channel2 8-bit right-aligned data holding register(DAC_DHR8R2)bit description

Bit	Access	Description
[31:8]	R	Reserved
[7：0]	RW	**DACC2DHR[7:0]:** DAC channel2 8-bit right-aligned data These bits are written by software which specify 8-bit data for DAC channel2.

12.5.9 Dual DAC 12-bit right-aligned data holding register(DAC_DHR12RD)

Register	Address offset	Access	Reset value	Description
DAC_DHR12RD	0x20	RW	0x0000_0000	Dual DAC 12-bit right-aligned data holding register

31	30	29	28	27	26	25	24
Reserved				DACC2DHR[11:8]			
23	22	21	20	19	18	17	16
DACC2DHR[7:0]							
15	14	13	12	11	10	9	8
Reserved				DACC1DHR[11:8]			
7	6	5	4	3	2	1	0
DACC1DHR[7:0]							

Dual DAC 12-bit right-aligned data holding register(DAC_DHR12RD)bit description

Bit	Access	Description
[31:28]	R	Reserved
[27：16]	RW	**DACC2DHR[11:0]:** DAC channel2 12-bit right-aligned data These bits are written by software which specify 12-bit data for DAC channel2.
[15:12]	R	Reserved
[11：0]	RW	**DACC1DHR[11:0]:** DAC channel1 12-bit right-aligned data These bits are written by software which specify 12-bit data for DAC channel1.

12.5.10 DUAL DAC 12-bit left aligned data holding register(DAC_DHR12LD)

Register	Address offset	Access	Reset value	Description
DAC_DHR12LD	0x24	RW	0x0000_0000	DUAL DAC 12-bit left aligned data holding register

31	30	29	28	27	26	25	24
DACC2DHR[11:4]							
23	22	21	20	19	18	17	16
DACC2DHR[3:0]				Reserved			
15	14	13	12	11	10	9	8
DACC1DHR[11:4]							
7	6	5	4	3	2	1	0
DACC1DHR[3:0]				Reserved			

DUAL DAC 12-bit left aligned data holding register(DAC_DHR12LD)bit description

Bit	Access	Description
[31：20]	RW	**DACC2DHR[11:0]:** DAC channel2 12-bit left-aligned data These bits are written by software which specify 12-bit data for DAC channel2.
[19:16]	R	Reserved
[15：4]	RW	**DACC1DHR[11:0]:** DAC channel1 12-bit left-aligned data These bits are written by software, which specifies 12-bit data for DAC channel1.
[3:0]	R	Reserved

12.5.11 DUAL DAC 8-bit right aligned data holding register(DAC_DHR8RD)

Register	Address offset	Access	Reset value	Description
DAC_DHR8RD	0x28	RW	0x0000_0000	DUAL DAC 8-bit right aligned data holding register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
DACC2DHR[7:0]							
7	6	5	4	3	2	1	0
DACC1DHR[7:0]							

DUAL DAC 8-bit right aligned data holding register(DAC_DHR8RD)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15：8]	RW	**DACC2DHR[15:8]:** DAC channel2 8-bit right-aligned data These bits are written by software which specify 8-bit data for DAC channel2.
[7：0]	RW	**DACC1DHR[7:0]:** DAC channel1 8-bit right-aligned data These bits are written by software which specify 8-bit data for DAC channel1.

12.5.12 DAC channel1 data output register(DAC_DOR1)

Register	Address offset	Access	Reset value	Description
DAC_DOR1	0x2C	RW	0x0000_0000	DAC channel1 data output register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				DACC1DOR[11:8]			
7	6	5	4	3	2	1	0
DACC1DOR[7:0]							

DAC channel1 data output register(DAC_DOR1)bit description

Bit	Access	Description
[31:12]	R	Reserved
[11：0]	RW	**DACC1DOR[11:0]:** DAC channel1 data output These bits are read only, they contain data output for DAC channel1.

12.5.13 DAC channel2 data output register(DAC_DOR2)

Register	Address offset	Access	Reset value	Description
DAC_DOR2	0x30	RW	0x0000_0000	DAC channel2 data output register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				DACC2DOR[11:8]			
7	6	5	4	3	2	1	0
DACC2DOR[7:0]							

DAC channel2 data output register(DAC_DOR2)bit description

Bit	Access	Description
[31:12]	R	Reserved,
[11：0]	RW	**DACC2DOR[11:0]:** DAC channel2 data output These bits are read only, they contain data output for DAC channel2.

13 Direct memory access controller（DMA）

13.1 DMA introduction

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

There are two DMAC in MG32F157xx : DMAC1 and DMAC2. DMAC1 has 7 hardware channels and DMAC2 has 5 hardware channels. All the channels share the AHB bus master bus interface. An arbitration scheme decides which of them is granted by the master bus interface.

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

13.2 DMA Characteristics

- 1 AHB master interface, 1 AHB slave interface.
- AMBA 2.0-compliant
- 1 AHB slave interface –used to program the EMAC
- 2 AHB master interface(s)
- 12 (7 for DMAC1, 5 for DMAC2) independent channels
- 12 (7 for DMAC1, 5 for DMAC2) hardware handshake signals
- Support for memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral DMA transfers
- Support for disabling channel without data loss
- Source and destination can be on different AHB layers
- Configurable data bus width for each AHB master interface
- Programmable source and destination for each channel
- Programmable DMA requesting source, priority for hardware handshaking interfaces, DMA handshake interfaces for each channel
- Configurable transfer type, transfer size for each channel
- Support single block DMA transfer
- Support DMAC as the flow controller of DMA transfer
- Support scatter and gather
- Support pseudo fly-by operation
- Support FIFO mode for each channel to improve the bandwidth

13.3 Pre-Configuration

To use the DMAC, the GPIO and AFIO should be configured correctly if needed. To save the power, user can turn off the peripheral clocks if not need. Before configuring the DMAC1, user should enable

the APB1 clock (SFR RCC_APB1PRE) and enable the DMAC clock (bit 15 and bit 0 of SFR RCC_APB1ENR). Before configuring the DMAC2, user should enable the APB2 clock (SFR RCC_APB2PRE) and enable the DMAC clock (bit 15 and bit 0 of SFR RCC_APB2ENR).

13.4 DMAC channel mapping

Tab 13.4-1 DMAC1 Channel list

Peripheral assembly 1	ADC1	TIM2_CH3	TIM4_CH1	QSPI0	AES_RX		
Peripheral assembly 2	UART3_TX	TIM1_CH1	TIM2_UP	TIM3_CH3	SPI1_RX	AES_TX	
Peripheral assembly 3	UART3_RX	TIM1_CH2	TIM3_CH4	TIM3_UP	SPI1_TX		
Peripheral assembly 4	UART1_TX	TIM1_CH4	TIM1_TRIG	TIM1_COM	TIM4_CH2	SPI2_RX	I2C2_TX
Peripheral assembly 5	UART1_RX	TIM1_UP	SPI2_TX	TIM2_CH1	TIM4_CH3	I2C2_RX	
Peripheral assembly 6	UART2_RX	TIM1_CH3	TIM3_CH1	TIM3_TRIG	I2C1_TX		
Peripheral assembly 7	UART2_TX	TIM2_CH2	TIM2_CH4	TIM4_UP	I2C1_RX		

	Peripheral assembly 1	Peripheral assembly 2	Peripheral assembly 3	Peripheral assembly 4	Peripheral assembly 5	Peripheral assembly 6	Peripheral assembly 7
ADC	ADC1						
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX		
UART		UART3_TX	UART3_RX	UART1_TX	UART1_RX	UART2_RX	UART2_TX
I2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP
QSPI	QSPI0						
AES	AES_RX	AES_TX					

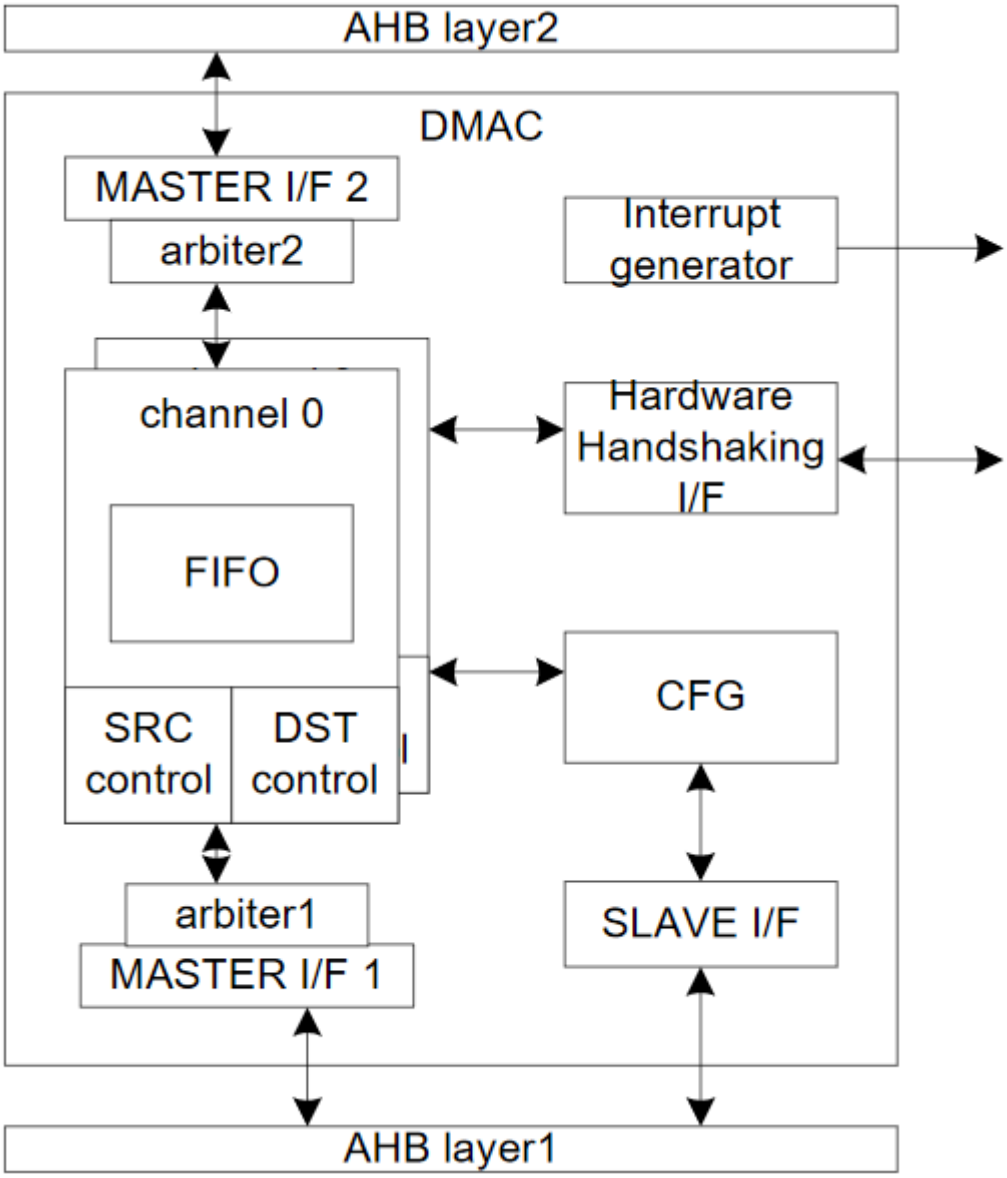
Tab 13.4-2 DMAC2 Channel list

Peripheral assembly 1	TIM5_CH4	TIM5_TRIG	TIM8_CH3	TIM8_UP	SPI3_RX	
Peripheral assembly 2	TIM8_CH4	TIM8_TRIG	TIM8_COM	TIM5_CH3	TIM5_UP	SPI3_TX
Peripheral assembly 3	TIM8_CH1	UART4_RX	TIM6_UP	DAC1		
Peripheral assembly 4	TIM5_CH2		TIM7_UP	DAC2		
Peripheral assembly 5	ADC3	TIM8_CH2	TIM5_CH1	UART4_TX		

	Peripheral assembly 1	Peripheral assembly 2	Peripheral assembly 3	Peripheral assembly 4	Peripheral assembly 5
ADC					ADC3
DAC			DAC1	DAC2	
SPI	SPI3_RX	SPI3_TX			
UART			UART4_RX		UART4_TX
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP		TIM5_CH2	TIM5_CH1
TIM6			TIM6_UP		
TIM7				TIM7_UP	
TIM8	TIM8_CH3 TIM8_UP	TIM8_CH4 TIM8_TRIG TIM8_COM	TIM8_CH1		TIM8_CH2

13.5 DMA Block Diagram

Fig 13.5-1 DMA block diagram in connectivity line devices



The DMAC including six main parts:

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

13.6 Function Overview

13.6.1 DMA operation

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

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

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

13.6.2 Arbitration for AHB master interface

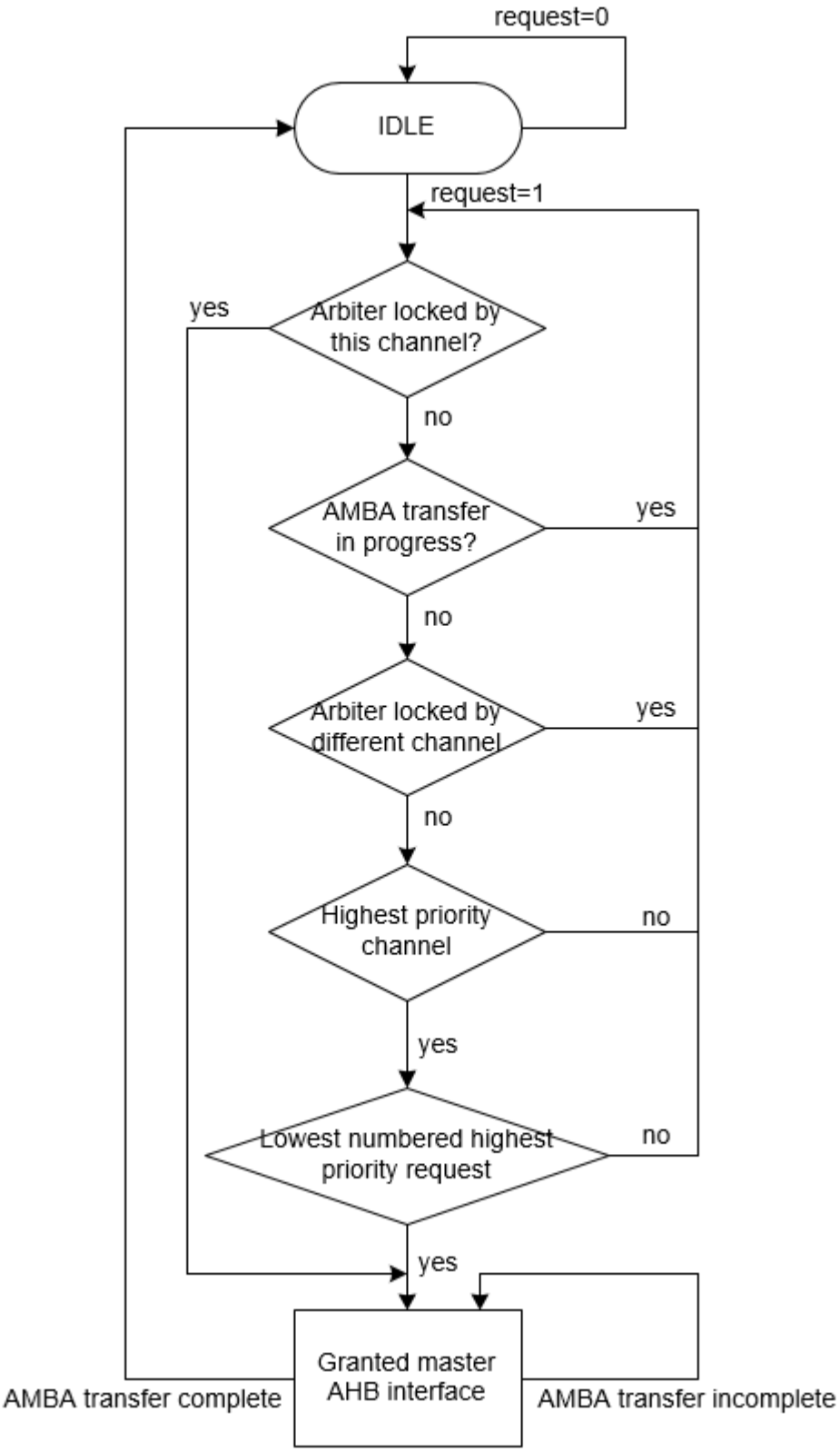
Each DAMC channel has two request lines that request ownership of a particular master bus interface: channel source and channel destination request lines. Source and destination arbitrate separately for the bus.

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

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

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

Fig 13.6-1 Arbitration Flow for Master Bus Interface



13.6.3 DMA Handshaking

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

Software Handshake

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

The software handshaking registers are:

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

13.6.4 Hardware Handshake

Each DMAC has 16 hardware handshake interfaces. The channel interfaces can be configured through the SFR CFGxL.HS_SEL_SRC, CFGxH.SRC_PER, CFGxL.HS_SEL_DST, CFGxH.DST_PER. The figure below shows the DMA single transfer waveform and DMA burst transfer waveform.

Fig 13.6-2 DMA single data transfer

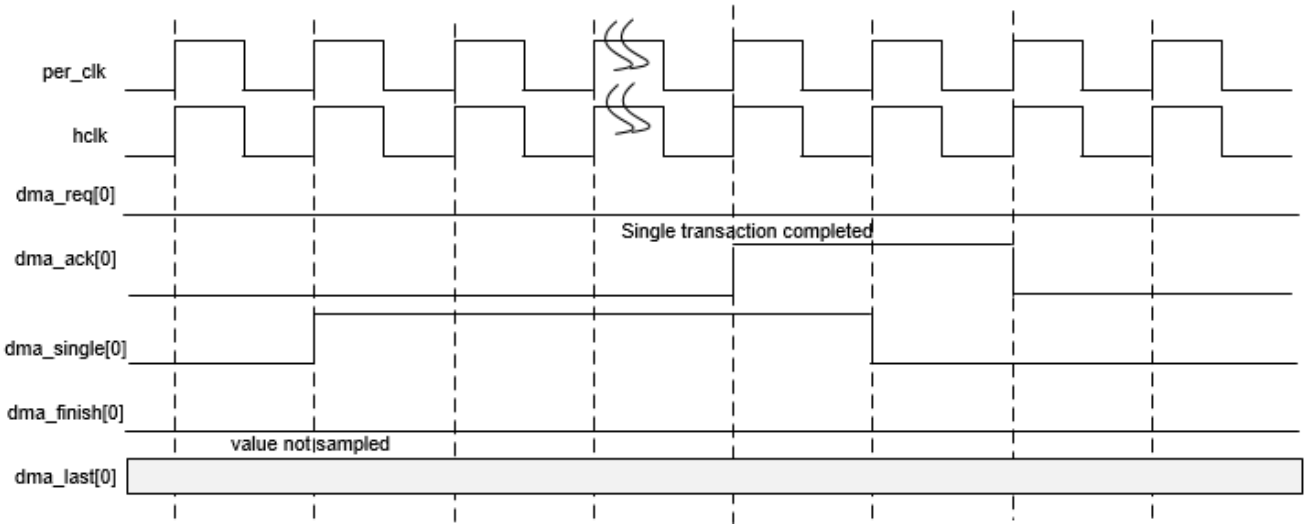
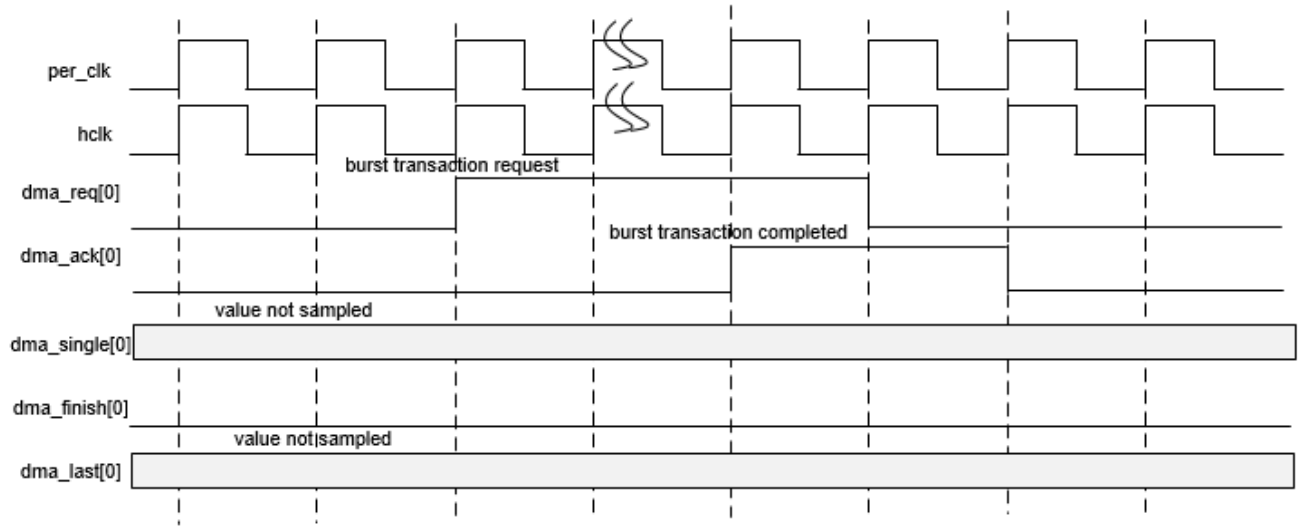


Fig 13.6-3 DMA burst data transfer



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

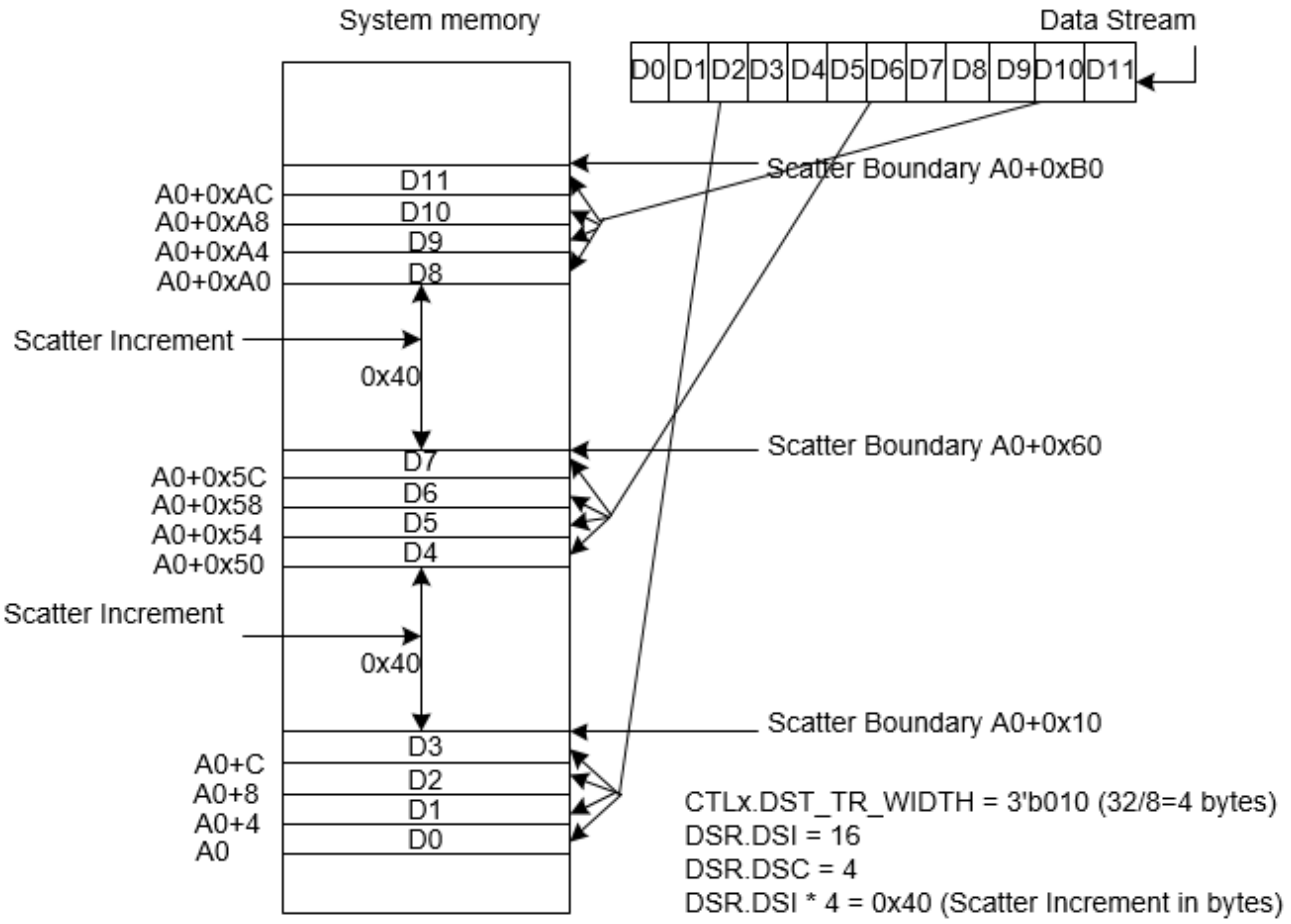
Note: *No handshaking interfaces are needed for the data transfer between DMA and SRAM.*

13.6.5 Scatter

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

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

Fig 13.6-4 Example of Destination Scatter Transfer

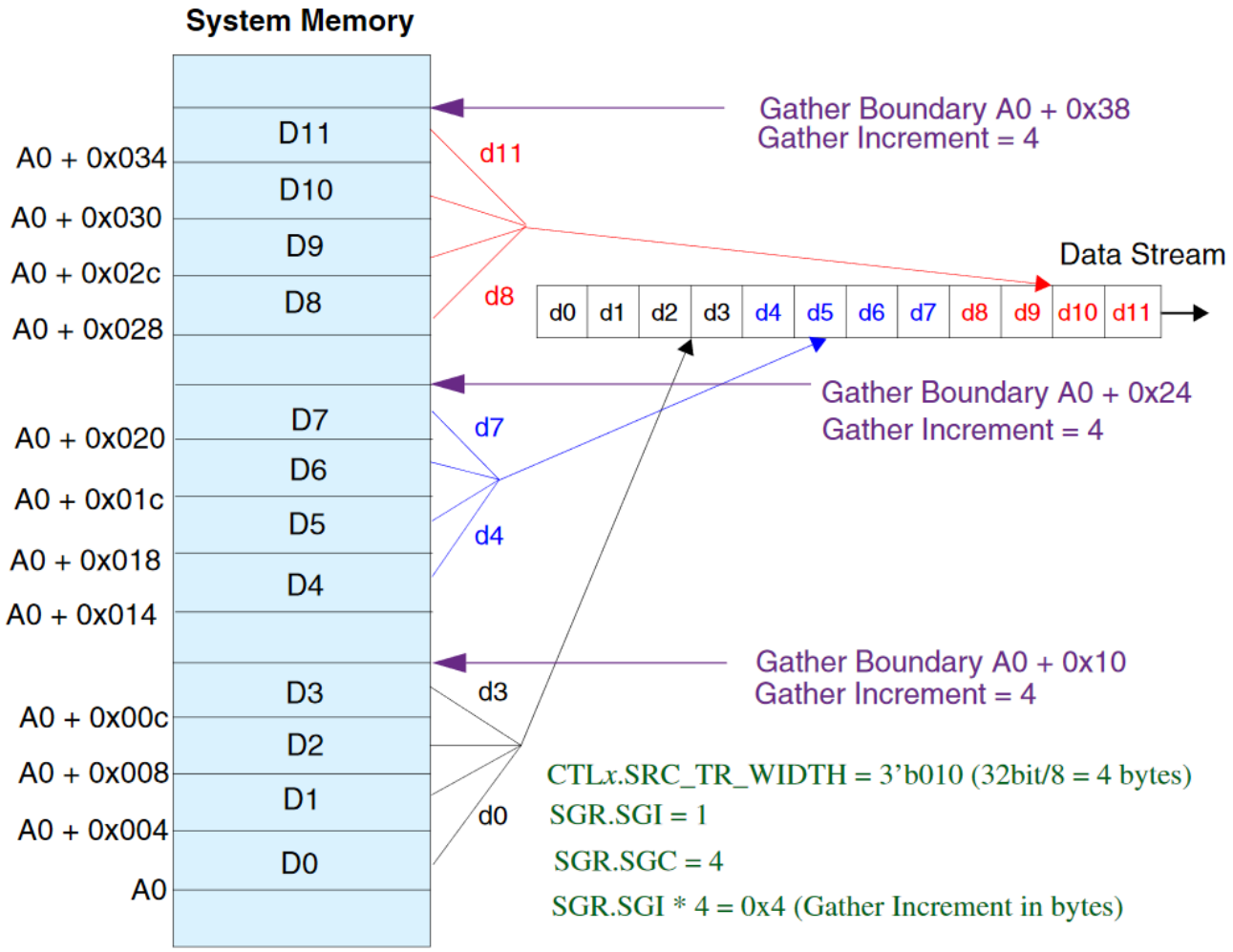


13.6.6 Gather

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

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

Fig 13.6-5 Source Gather when SGR.SGI=0x1



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

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

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

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

13.6.7 AHB Transfer Error Handling

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

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

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

The contents of the FIFO are not cleared, but they become inaccessible and are overwritten once the

channel is re-enabled to start a new sequence.

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

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

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

13.6.8 Disabling a Channel Prior to Transfer Completion

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

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

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

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

13.6.9 Programming Examples

Below is an example showing how to program DMA.

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

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

13.6.10 Interrupt

Each Channel has five interrupt sources:

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

There are several groups of interrupt-related registers:

- RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr

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

13.7 DMA Register

Tab 13.7-1 DMA register map

Offset	Register	Reset value	Description
DMA1：DMA1_BA = 0x4002_0000 DMA2：DMA2_BA = 0x4002_0400			
0x0+x*0x58	DMA_SARx (x=0...6)	0x0000_0000	Source Address for Channel x
0x8+x*0x58	DMA_DARx (x=0...6)	0x0000_0000	Destination Address for Channel x
0x18+x*0x58	DMA_CTLLx (x=0...6)	0x0030_4801	Channel Control register Low
0x1C+x*0x58	DMA_CTLHx (x=0...6)	0x0000_0002	Channel Control register High
0x40+x*0x58	DMA_CFGLx (x=0...6)	0x00000C00+x*0x20	Channel Configuration register Low
0x44+x*0x58	DMA_CFGHx (x=0...6)	0x0000_0002	Channel Configuration register High
0x48+x*0x58	DMA_SGRx (x=0...6)	0x0000_0000	Source Gather Register
0x50+x*0x58	DMA_DSRx (x=0...6)	0x0000_0000	Destination Scatter Register
Interrupt Raw Status Register(DMA_INTRAW)			
0x2C0	RawTfr	0x0000_0000	End of DMA transfer interrupts the original status register
0x2C8	RawBlock	0x0000_0000	The end of block transfer interrupts the original status register
0x2D0	RawSrcTran	0x0000_0000	The source end of transmission interrupts the original status register
0x2D8	RawDstTran	0x0000_0000	Destination end of transmission interrupt original status register
0x2E0	RawErr	0x0000_0000	Bus error interrupt register
Interrupt Status Register(DMA_INTSTA)			
0x2E8	StatusTfr	0x0000_0000	End of DMA transfer interrupts the original status register
0x2F0	StatusBlock	0x0000_0000	The end of block transfer interrupts the original status register
0x2F8	StatusSrcTran	0x0000_0000	The source end of transmission interrupts the original status register
0x300	StatusDstTran	0x0000_0000	Destination end of transmission interrupt original status register
0x308	StatusErr	0x0000_0000	Bus error interrupt register
DMA interrupt mask register group(DMA_INTMSK)			
0x310	MaskTfr	0x0000_0000	DMA transfer end interrupt mask register
0x318	MaskBlock	0x0000_0000	Block transfer end interrupt mask register
0x320	MaskSrcTran	0x0000_0000	Source-end transfer interrupt mask register
0x328	MaskDstTran	0x0000_0000	Destination end of transmission interrupt mask register
0x330	MaskErr	0x0000_0000	Bus error interrupt mask register
DMA interrupt clears register group(DMA_INTCLR)			
0x338	ClrTfr	0x0000_0000	DMA transfer end interrupt clear register
0x340	ClrBlock	0x0000_0000	Block transfer end interrupt clear register
0x348	ClrSrcTran	0x0000_0000	Source-end transfer interrupt clear register
0x350	ClrDstTran	0x0000_0000	Destination end of transmission interrupt clear register
0x358	ClrErr	0x0000_0000	Bus error interrupt clear register
0x360	DMA_StatusInt	0x0000_0000	Status for each Interrupt type: StatusInt
0x368	DMA_ReqSrcReg	0x0000_0000	Source Software Transaction Request register
0x370	DMA_ReqDstReg	0x0000_0000	Destination Software Transaction Request register

(continued)

Offset	Register	Reset value	Description
0x378	DMA_SglRqSrcReg	0x0000_0000	Source Single Transaction Request register
0x380	DMA_SglRqDstReg	0x0000_0000	Destination Single Transaction Request register
0x388	DMA_LstSrcReg	0x0000_0000	Source Last Transaction Request register
0x390	DMA_LstDstReg	0x0000_0000	Destination Last Transaction Request register
0x398	DMA_DmaCfgReg	0x0000_0000	DMA Configuration Register
0x3A0	DMA_ChEnReg	0x0000_0000	Channel Enable register
0x3F8	DMA_DmaCompsID0	0x4457_1110	DMA ID register0
0x3FC	DMA_DmaCompsID1	0x0000_0000	DMA ID register1

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

- *TIM2_COM and TIM2_TRIG*
- *TIM3_CC2 and TIM3_COM*
- *TIM4_CC4, TIM4_COM and TIM4_TRIG*

2.ADC only support single DMA transfer

13.7.1 Source Address for Channel x(DMA_SARx)(x = 0...6)

Register	Address offset	Access	Reset value	Description
DMA_SARx	0x000 + 0x058 * x	RW	0x0000_0000	Source Address for Channel x

Source Address for Channel x bit description

Bit	Access	Description
[31:0]	RW	**SAR:** The source address of the current transmission is automatically updated after the source end completes a read. The update mode is determined by the CTL register SINC.

This register is software-writable when corresponding to CH_EN=0.

The initial value of the SAR is usually configured by the software before opening CH_EN, or updated automatically by the LLI before a new DMA transfer, and updated in real time by the hardware state machine when the DMA transfer has begun.

13.7.2 Destination Address for Channel x(DMA_DARx)(x = 0...6)

Register	Address offset	Access	Reset value	Description
DMA_DARx	0x008 + 0x058 * x	RW	0x0000_0000	Destination Address for Channel x

Destination Address for Channel x(DMA_DARx)bit description

Bit	Access	Description
[31:0]	RW	**DAR:** The source address of the current transmission is automatically updated after the source end completes a read. The update mode is determined by the CTL register SINC.

This register is software-writable when corresponding to CH_EN=0.

The initial value of the DAR is also usually configured by the software before opening CH_EN, or updated automatically by the LLI before a new DMA transfer, and updated in real time by the hardware state machine when the DMA transfer has begun.

In continuous multi-block transmission, there may be an address mismatch between the end of the block and the start of the block. At this time, DMA will fine-tune SAR/DAR. For example, when 9 blocks are transmitted, the data transmission bit width of the source side is 16, the data transmission bit width of the target side is 32, and the target transmission mode is continuous multi-block. In this way, the target end needs to write 9x16/32=4.5 times each time. If the address is incremental, DMA automatically writes low 16 at the end of the first round and high 16 at the beginning of the next round. If the address is decremented, high 16 is written at the end of the first round and low 16 is written at the beginning of the next round.

13.7.3 Channel Control register Low(DMA_CTLLx)(x = 0...6)

Register	Address offset	Access	Reset value	Description
DMA_CTLLx	0x018 + 0x058 * x	RW	0x0030_4801	Channel Control register Low

31	30	29	28	27	26	25	24
Reserved			LLP_SRC_EN	LLP_DST_EN	Reserved		
23	22	21	20	19	18	17	16
Reserved	TT_FC[2:0]			Reserved			SRC_MSIZE[2]
15	14	13	12	11	10	9	8
SRC_MSIZE[1:0]		DEST_MSIZE[2:0]			SINC[1:0]		DINC[1]
7	6	5	4	3	2	1	0
DINC[0]	SRC_TR_WIDTH[2:0]			DST_TR_WIDTH[2:0]			INT_EN

Channel Control register Low(DMA_CTLLx)bit description

Bit	Access	Description
[31:29]	R	Reserved
[28]	RW	**LLP_SRC_EN** Source link mode switch. 0: disable. 1: Open; After this function is enabled, related configurations on the source end are automatically read and loaded by LLI (start address LLP->LOC) after block transmission is complete. This mode does not take effect when the LLP->LOC register is 0.
[27]	RW	**LLP_DST_EN** Link mode switch of the target end. 0: disable. 1: Open; After this function is enabled, the configurations of the target end are automatically read and loaded by LLI (start address LLP->LOC) after block transmission is complete. This mode does not take effect when the LLP->LOC register is 0.
[26:23]	RW	Reserved,Fixed with 0.
[22:20]	RW	**TT_FC[2:0]:** Transmission type selection, multi-module transmission of the register must be the same configuration 0: MEM -> MEM; 1: MEM - > peripherals; 2: Peripheral ->MEM; 3: Peripheral -> Peripheral; 4-7: reserved;
[19:17]	-	Reserved
[16:14]	RW	**SRC_MSIZE[2:0]:** Source Burst Transaction Length. Multiplied by SRC_TR_WIDTH is the total number of data read per transfer, 0x0: Number of data items to be transferred is 1 0x1: Number of data items to be transferred is 4 0x2: Number of data items to be transferred is 8 0x3: Number of data items to be transferred is 16 0x4: Number of data items to be transferred is 32 0x5: Number of data items to be transferred is 64 0x6: Number of data items to be transferred is 128 0x7: Number of data items to be transferred is 256

[13:11]	RW	**DEST_MSIZE[2:0]:** The length configuration of a single target transmission, multiplied by DST_TR_WIDTH, is the total number of data written out per transmission 0x0: Number of data items to be transferred is 1 0x1: Number of data items to be transferred is 4 0x2: Number of data items to be transferred is 8 0x3: Number of data items to be transferred is 16 0x4: Number of data items to be transferred is 32 0x5: Number of data items to be transferred is 64 0x6: Number of data items to be transferred is 128 0x7: Number of data items to be transferred is 256
[10:9]	RW	**SINC[1:0]:** Source Address Increment. 0x0: Increments the source address 0x1: Decrements the source address 0x2: No change in the source address 0x3: No change in the source address
[8:7]	RW	**DINC[1:0]:** Destination Address Increment. 0x0: Increments the destination address 0x1: Decrements the destination address 0x2: No change in the destination address 0x3: No change in the destination address
[6:4]	RW	**SRC_TR_WIDTH[2:0]:** Source Transfer Width 0x0: Source transfer width is 8 bits 0x1: Source transfer width is 16 bits 0x2: Source transfer width is 32 bits 0x3: Source transfer width is 32 bits 0x4: Source transfer width is 32 bits 0x5: Source transfer width is 32 bits 0x6: Source transfer width is 32 bits 0x7: Source transfer width is 32 bits
[3:1]	RW	**DST_TR_WIDTH[2:0]:** Destination Transfer Width 0x0: Destination transfer width is 8 bits 0x1: Destination transfer width is 16 bits 0x2: Destination transfer width is 32 bits 0x3: Destination transfer width is 32 bits 0x4: Destination transfer width is 32 bits 0x5: Destination transfer width is 32 bits 0x6: Destination transfer width is 32 bits 0x7: Destination transfer width is 32 bits
[0]	RW	**INT_EN:** Interrupt Enable Bit. 0: off, 1: on. Note: The Raw interrupt register is not controlled by this enable;

When CH_EN=0, the software can be written.

13.7.4 Channel Control register High(DMA_CTLHx)(x = 0...6)

Register	Address offset	Access	Reset value	Description
DMA_CTLHx	0x01C + 0x058 * x	RW	0x0000_0002	Channel Control register High

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

Channel Control register High(DMA_CTLHx)bit description

Bit	Access	Description
[31:13]	R	Reserved
[12]	RW	**DONE:** Current LLI completion status, 0: not completed; 1: complete; When the software is configuring the LLI, the bit must be set to 0. After starting the LLI, the bit can be observed to find the completion status.
[11:0]	RW	**BLOCK_TS[11:0]:** W:The total length of a transfer block is equal to BLOCK_TS * SRC_TR_WIDTH R:The value is retained before startup. After startup, the length is actually moved from the source end, increasing from 0

When CH_EN=0, the software can be written.

When the status write back function is turned on, the register is written to the LLI address when the last transfer block DONE is pulled up.

The software can detect LLI->CTLx_H->DONE to determine whether the block transfer has ended, and the DONE must be cleared before the Channel is enabled

13.7.5 Channel Configuration register Low(DMA_CFGLx)(x = 0...6)

Register	Address offset	Access	Reset value	Description
DMA_CFGLx	0x040 + 0x058 * x	RW	0x0000_0C00 + 0x20 * x	Channel Configuration register Low

31	30	29	28	27	26	25	24
RELOAD_DST	RELOAD_SRC	MAX_ABRST[9:4]					
23	22	21	20	19	18	17	16
MAX_ABRST[3:0]				SRC_HS_POL	DST_HS_POL	Reserved	
15	14	13	12	11	10	9	8
Reserved				HS_SEL_SRC	HS_SEL_DST	FIFO_EMPTY	CH_SUSP
7	6	5	4	3	2	1	0
CH_PRIOR			Reserved				

Channel Configuration register Low(DMA_CFGLx)bit description

Bit	Access	Description
[31]	RW	**RELOAD_DST:** Automatic loading DAR is enabled. 0: disabled. 1: Open; In multi-module transmission, DMA can automatically load new DAR when a transmission block is finished.

[30]	RW	**RELOAD_SRC:** Automatic loading SRC is enabled. 0: disabled. 1: Open; In multi-module transmission, DMA can automatically load new SRC when a transmission block is finished.
[29:20]	RW	**MAX_ABRST:** Maximum single transmission length upper limit, 0: do not do the upper limit, generally equal to the FIFO depth
[19]	RW	**SRC_HS_POL:** Source Handshaking Interface Polarity. 0x0: Source Handshaking Interface Polarity is Active high 0x1: Source Handshaking Interface Polarity is Active low
[18]	RW	**DST_HS_POL:** Destination Handshaking Interface Polarity. 0x0: Destination Handshaking Interface Polarity is Active high 0x1: Destination Handshaking Interface Polarity is Active low
[17:12]	R	Reserved
[11]	RW	**HS_SEL_SRC:** Source handshake selection 0x0: Hardware handshaking interface. 0x1: Software handshaking interface. The MEM type does not require a handshake;
[10]	RW	**HS_SEL_DST:** Destination terminal selection 0x0: Hardware handshaking interface. 0x1: Software handshaking interface. The MEM type does not require a handshake;
[9]	R	**FIFO_EMPTY:** FIFO data observation signal; 0x0: Channel FIFO is not empty 0x1: Channel FIFO is empty
[8]	RW	**CH_SUSP:** The Channel pause function is enabled. 0: disables. 1: Pause the transmission; It is best to operate the pause function when FIFO_EMPTY is 1;
[7:5]	RW	**CH_PRIOR:** Channel priority: When multiple channels request the bus at the same time, for DMA1, 6 has the highest priority and 0 has the lowest priority. For DMA2,4 has the highest priority and 0 has the lowest priority.
[4:0]	R	Reserved

13.7.6 Channel Configuration register High(DMA_CFGHx)(x = 0...6)

Register	Address offset	Access	Reset value	Description
DMA_CFGHx	0x044 + 0x058 * x	RW	0x0000_0002	Channel Configuration register High

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	DEST_PER[3:0]				SRC_PER[3:1]		
7	6	5	4	3	2	1	0
SRC_PER[0]	SS_UPD_EN	DS_UPD_EN	PROTCTL[2:0]			FIFO_MODE	Reserved

Channel Configuration register High(DMA_CFGHx)bit description

Bit	Access	Description
[31:15]	R	Reserved

[14:11]	RW	**DEST_PER[3:0]:** Select the target hardware handshake signal.It takes effect when HS_SEL_DST=0 **DMA1** 0:ADC1,TIM2_CH3,TIM4_CH1,QSPI0,AES_RX; 1:UART3_TX,TIM1_CH1,TIM2_UP,TIM3_CH3,SPI1_RX,AES_TX; 2:UART3_RX,TIM1_CH2,TIM3_CH4,TIM3_UP,SPI1_TX,AES_TX; 3:UART1_TX,TIM1_CH4,TIM1_TRIG,TIM1_COM,TIM4_CH2,SPI2_RX,I2C2_TX; 4:UART1_RX,TIM1_UP,SPI2_TX,TIM2_CH1,TIM4_CH3,I2C2_RX, AES_RX; 5:UART2_RX,TIM1_CH3,TIM3_CH1,TIM3_TRIG,I2C1_TX; 6:UART2_TX,TIM2_CH2,TIM2_CH4,TIM4_UP,I2C1_RX; **DMA2** 0: TIM5_CH4,TIM5_TRIG,TIM8_CH3,TIM8_UP,SPI3_RX; 1: TIM8_CH4,TIM8_TRIG,TIM8_COM,TIM5_CH3,TIM5_UP,SPI3_TX; 2: TIM8_CH1,UART4_RX,TIM6_UP,DAC1; 3: TIM5_CH2,TIM7_UP,DAC2; 4: ADC3,TIM8_CH2,TIM5_CH1,UART4_TX;
[10:7]	RW	**SRC_PER[3:0]:** Select the target hardware handshake signal.It takes effect when HS_SEL_SRC==0 **DMA1** 0:ADC1,TIM2_CH3,TIM4_CH1,QSPI0,AES_RX; 1:UART3_TX,TIM1_CH1,TIM2_UP,TIM3_CH3,SPI1_RX,AES_TX; 2:UART3_RX,TIM1_CH2,TIM3_CH4,TIM3_UP,SPI1_TX,AES_TX; 3:UART1_TX,TIM1_CH4,TIM1_TRIG,TIM1_COM,TIM4_CH2,SPI2_RX,I2C2_TX; 4:UART1_RX,TIM1_UP,SPI2_TX,TIM2_CH1,TIM4_CH3,I2C2_RX,AES_RX; 5:UART2_RX,TIM1_CH3,TIM3_CH1,TIM3_TRIG,I2C1_TX; 6:UART2_TX,TIM2_CH2,TIM2_CH4,TIM4_UP,I2C1_RX; **DMA2** 0: TIM5_CH4,TIM5_TRIG,TIM8_CH3,TIM8_UP,SPI3_RX; 1:TIM8_CH4,TIM8_TRIG,TIM8_COM,TIM5_CH3,TIM5_UP,SPI3_TX; 2:TIM8_CH1,UART4_RX,TIM6_UP,DAC1; 3:TIM5_CH2,TIM7_UP,DAC2; 4:ADC3,TIM8_CH2,TIM5_CH1,UART4_TX;
[6]	RW	**SS_UPD_EN:** Status update on the source end is enabled. 0: disables. 1: After the SSTAT is updated, it is also written to LLI.
[5]	RW	**DS_UPD_EN:** Status update of the target end is enabled. 0: disables. 1: After the SSTAT is updated, it is also written to LLI.
[4:2]	RW	Reserved, Keep 1;
[1]	RW	**FIFO_MODE:** FIFO Mode Select. 0: Start reading every time the write is complete; 1: Read only when the state is half full;
[0]	-	Reserved

13.7.7 Source Gather Register(DMA_SGRx)(x = 0...6)

Register	Address offset	Access	Reset value	Description
DMA_SGRx	0x048 + 0x058 * x	RW	0x0000_0000	Source Gather Register

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

Source Gather Register(DMA_SGRx)bit description

Bit	Access	Description
[31:20]	RW	**SGC[11:0]:** Source Gather Count. Source contiguous tranfer count between successive gather boundaries.
[19:0]	RW	**SGI[19:0]:** Source Gather Interval.

13.7.8 Destination Scatter Register(DMA_DSRx)(x = 0...6)

Register	Address offset	Access	Reset value	Description
DMA_DSRx	0x050 + 0x058 * x	RW	0x0000_0000	Destination Scatter Register

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

Destination Scatter Register(DMA_DSRx)bit description

Bit	Access	Description
[31:20]	RW	**DSC[11:0]:** Destination Scatter Count. Destination contiguous transfer count between successive scatter boundaries.
[19:0]	RW	**DSI[19:0]:** Destination Scatter Interval.

13.7.9 Interrupt Raw Status Register(DMA_INTRAW)

Register	Address offset	Access	Reset value
RawTfr	0x2C0	R	0x0000_0000
RawBlock	0x2C8	R	0x0000_0000
RawSrcTran	0x2D0	R	0x0000_0000
RawDstTran	0x2D8	R	0x0000_0000
RawErr	0x2E0	R	0x0000_0000

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

Interrupt Raw Status Register(DMA_INTRAW)bit description

Bit	Access	Description
[31:7]	R	Reserved
[6:0]	RW	**RawTfr / RawBlock / RawSrcTran /RawDstTran / RawErr:** The last source end transmission ends the interrupt information, when the interrupt occurs, pull up; Each bit represents the corresponding CH, which can be cleared by writing 1 for each bit in the corresponding ClearTfr. Bit6:CH7 ... Bit0:CH1 RAW information cannot be masked

This register can be modified directly, but this operation is not recommended, you should use the Clear* register to clear both Raw* and Status*.

13.7.10 Interrupt Status Register(DMA_INTSTA)

Register	Address offset	Access	Reset value
StatusTfr	0x2E8	R	0x0000_0000
StatusBlock	0x2F0	R	0x0000_0000
StatusSrcTran	0x2F8	R	0x0000_0000
StatusDstTran	0x300	R	0x0000_0000
StatusErr	0x308	R	0x0000_0000

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

Interrupt Status Register(DMA_INTSTA)bit description

Bit	Access	Description
[31:7]	R	Reserved

[6:0]	R	**StatusTfr / StatusBlock / StatusSrcTran / StatusDstTran / StatusErr:** All transmission ends interrupt information, when the interruption occurs pull up; Each bit represents the corresponding CH, which can be cleared by writing 1 for each bit in the corresponding ClearTfr. Bit6:CH7 ... Bit0:CH1 The register can be masked

13.7.11 Interrupt Mask Register(DMA_INTMSK)

Register	Address offset	Access	Reset value
MaskTfr	0x310	RW	0x0000_0000
MaskBlock	0x318	RW	0x0000_0000
MaskSrcTran	0x320	RW	0x0000_0000
MaskDstTran	0x328	RW	0x0000_0000
MaskErr	0x330	RW	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	WriteEN						
7	6	5	4	3	2	1	0
Reserved	MaskTfr / MaskBlock / MaskSrcTran / MaskDstTran / MaskErr						

Interrupt Mask Register(DMA_INTMSK)bit description

Bit	Access	Description
[31:15]	-	Reserved
[14:8]	W	**WriteEN:** The write function of each (MaskTfr / MaskBlock / MaskSrcTran / MaskDstTran / MaskErr) is enabled. 0: cannot be written. 1: writable; Bit14:CH7 ... Bit8:CH1
[7]	-	Reserved
[6:0]	RW	**MaskTfr / MaskBlock / MaskSrcTran / MaskDstTran / MaskErr:** Mask enable . 0: mask, 1: not mask. Each bit represents the corresponding CH; Bit6:CH7 ... Bit0:CH1

13.7.12 Interrupt Clear Register(DMA_INTCLR)

Register	Address offset	Access	Reset value
ClrTfr	0x338	W	0x0000_0000
ClrBlock	0x340	W	0x0000_0000
ClrSrcTran	0x348	W	0x0000_0000
ClrDstTran	0x350	W	0x0000_0000
ClrErr	0x358	W	0x0000_0000

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

Interrupt Clear Register(DMA_INTCLR)bit description

Bit	Access	Description
[31:7]	R	Reserved
[6:0]	W	**ClrTfr / ClrBlock / ClrSrcTran / ClrDstTran / ClrErr:** Write 1 to clear RawTfr and StatusTfr information, Each bit represents the corresponding CH; Bit6:CH7 ... Bit0:CH1

13.7.13 Status for each Interrupt type(DMA_StatusInt)

Register	Address offset	Access	Reset value	Description
DMA_StatusInt	0x360	R	0x0000_0000	Status for each Interrupt type

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

Status for each Interrupt type(DMA_StatusInt)bit description

Bit	Access	Description
[31:5]	R	Reserved

[4]	R	**ERR:** OR of the contents of StatusErr 0x0 (INACTIVE): OR of the contents of StatusErr register is 0 0x1 (ACTIVE): OR of the contents of StatusErr register is 1
[3]	R	**DSTT:** OR of the contents of StatusDstTran 0x0 (INACTIVE): OR of the contents of StatusDstTran register is 0 0x1 (ACTIVE): OR of the contents of StatusDstTran register is 1
[2]	R	**SRCT:** OR of the contents of StatusSrcTran 0x0 (INACTIVE): OR of the contents of StatusSrcTran register is 0 0x1 (ACTIVE): OR of the contents of StatusSrcTran register is 1
[1]	R	**BLOCK:** OR of the contents of StatusBlock register 0x0 (INACTIVE): OR of the contents of StatusBlock register is 0 0x1 (ACTIVE): OR of the contents of StatusBlock register is 1
[0]	R	**TFR:** OR of the contents of StatusTfr register 0x0 (INACTIVE): OR of the contents of StatusTfr register is 0 0x1 (ACTIVE): OR of the contents of StatusTfr register is 1

13.7.14 Source Software Transaction Request register(DMA_ReqSrcReg)

Register	Address offset	Access	Reset value	Description
DMA_ReqSrcReg	0x368	RW	0x0000_0000	Source Software Transaction Request register

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

Source Software Transaction Request register(DMA_ReqSrcReg)bit description

Bit	Access	Description
[31:15]	-	Reserved
[14:8]	W	**WriteEN:** The write function of each SRC_REQ is enabled. 0: cannot be written. 1: writable; Bit14:CH7 ... Bit8:CH1
[7]	-	Reserved
[6:0]	RW	**SRC_REQ :** Source handshake request: 0: do not apply. 1: Request a handshake; Each bit represents the corresponding CH, and HS_SEL_SRC of the corresponding CH must be 1 (software handshake) to take effect. Bit6:CH7 ... Bit0:CH1

13.7.15 Destination Software Transaction Request register(DMA_ReqDstReg)

Register	Address offset	Access	Reset value	Description
DMA_ReqDstReg	0x370	RW	0x0000_0000	Destination Software Transaction Request register

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

Destination Software Transaction Request register(DMA_ReqDstReg)bit description

Bit	Access	Description
[31:15]	-	Reserved
[14:8]	W	**WriteEN:** The write function of each DST_REQ is enabled. 0: cannot be written. 1: writable; Bit14:CH7 ... Bit8:CH1
[7]	-	Reserved
[6:0]	RW	**DST_REQ :** The target end makes a handshake request. 0: does not apply. 1: Request a handshake; Each bit represents the corresponding CH, and HS_SEL_DST of the corresponding CH must be 1 (software handshake) to take effect. Bit6:CH7 ... Bit0:CH1

13.7.16 Source Single Transaction Request register(DMA_SglRqSrcReg)

Register	Address offset	Access	Reset value	Description
DMA_SglRqSrcReg	0x378	RW	0x0000_0000	Source Single Transaction Request register

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

Source Single Transaction Request register(DMA_SglRqSrcReg)bit description

Bit	Access	Description
[31:15]	-	Reserved

[14:8]	W	**WriteEN:** The write function of each SRC_SGLREQ is enabled. 0: cannot be written. 1: writable; Bit14:CH7 ... Bit8:CH1
[7]	-	Reserved
[6:0]	RW	**SRC_SGLREQ:** Single handshake request at the source end. 0: do not apply. 1: Request a handshake; Each bit represents the corresponding CH, and HS_SEL_SRC of the corresponding CH must be 1 (software handshake) to take effect. Bit6:CH7 ... Bit0:CH11

13.7.17 Destination Single Transaction Request register(DMA_SglRqDstReg)

Register	Address offset	Access	Reset value	Description
DMA_SglRqDstReg	0x380	RW	0x0000_0000	Destination Single Transaction Request register

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

Destination Single Transaction Request register(DMA_SglRqDstReg)bit description

Bit	Access	Description
[31:15]	-	Reserved
[14:8]	W	**WriteEN:** For each DST_SGLREQ write enable, 0: cannot write. 1: writable; Bit14:CH7 ... Bit8:CH1
[7]	-	Reserved
[6:0]	RW	**DST_SGLREQ:** The target end requests a single handshake. 0: does not apply. 1: Request a handshake; Each bit represents the corresponding CH, and HS_SEL_DST of the corresponding CH must be 1 (software handshake) to take effect. Bit6:CH7 ... Bit0:CH1

13.7.18 Source Last Transaction Request register(DMA_LstSrcReg)

Register	Address offset	Access	Reset value	Description
DMA_LstSrcReg	0x388	RW	0x0000_0000	Source Last Transaction Request register

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

Source Last Transaction Request register(DMA_LstSrcReg)bit description

Bit	Access	Description
[31:15]	-	Reserved
[14:8]	W	**WriteEN:** For each LSTSRC_REQ write enable, 0: cannot write. 1: writable; Bit14:CH7 ... Bit8:CH1
[7]	-	Reserved
[6:0]	RW	**LSTSRC_REQ:** For the last handshake request at the source end, 0: do not apply. 1: Request a handshake; Each bit represents the corresponding CH, and HS_SEL_SRC of the corresponding CH must be 1 (software handshake) to take effect. Bit6:CH7 ... Bit0:CH1

13.7.19 Destination Last Transaction Request register(DMA_LstDstReg)

Register	Address offset	Access	Reset value	Description
DMA_LstDstReg	0x390	RW	0x0000_0000	Destination Last Transaction Request register

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

Destination Last Transaction Request register(DMA_LstDstReg)bit description

Bit	Access	Description
[31:15]	-	Reserved
[14:8]	W	**WriteEN:** For each LSTDST_REQ write enable, 0: cannot write. 1: writable; Bit14:CH7 ... Bit8:CH1
[7]	-	Reserved

[6:0]	RW	**LSTDST_REQ:** The last handshake request of the target end, 0: do not apply; 1: Request a handshake; Each bit represents the corresponding CH, and HS_SEL_DST of the corresponding CH must be 1 (software handshake) to take effect. Bit6:CH7 ... Bit0:CH1

13.7.20 DMA Configuration Register(DMA_DmaCfgReg)

Register	Address offset	Access	Reset value	Description
DMA_DmaCfgReg	0x398	RW	0x0000_0000	DMA Configuration Register

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

DMA Configuration Register(DMA_DmaCfgReg)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	RW	**DMA_EN:** DW_ahb_dmac Enable bit. 0x0 (DISABLED): DMAC Disabled 0x1 (ENABLED): DMAC Enabled When write 0 is turned off, it will be lowered according to the working status of all channels until all are finished.

13.7.21 Channel Enable register(DMA_ChEnReg)

Register	Address offset	Access	Reset value	Description
DMA_ChEnReg	0x3A0	RW	0x0000_0000	Channel Enable register

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

Channel Enable register(DMA_ChEnReg)bit description

Bit	Access	Description
[31:15]	R	Reserved
[14:8]	W	**CH_EN_WE:** The corresponding update of each CH_EN is enabled. The corresponding CH_EN can be changed only when the value is 1. Bit14:CH7 … Bit8:CH1
[7]	R	Reserved
[6:0]	RW	**CH_EN:** The enable switch of each Channel is turned on by write 1, and then pulled down after the hardware is completed. Bit6:CH7 … Bit0:CH1

This register can only be changed if DMA_EN (DMA CFG bit0) is 1, and if DMA_EN is 0, the register is fixed to 0.

CH_EN_WE allows software to modify CH_EN without first reading and then writing it. For example, when CH0 is opened, the CH_en_we can be directly written to 0x101, and the EN of other CH is not modified.

13.7.22 DMA ID register0(DMA_DmaCompsID0)

Register	Address offset	Access	Reset value	Description
DMA_DmaCompsID0	0x3F8	R	0x4457_1110	DMA ID register0

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

DMA ID register0(DMA_DmaCompsID0)bit description

Bit	Access	Description
[31:0]	R	**COMP_TYPE[31:0]:** DMA Component Type Number = ‘h44571110.

13.7.23 DMA ID register1(DMA_DmaCompsID1)

Register	Address offset	Access	Reset value	Description
DMA_DmaCompsID1	0x3FC	R	0x0000_0000	DMA ID register1

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

DMA ID register1(DMA_DmaCompsID1)bit description

Bit	Access	Description
[31:0]	R	**COMP_VER[31:0]:** DMA Component Version

14 Advanced-control timers (TIM1 and TIM8)

14.1 TIM1 and TIM8 introduction

The advanced-control timers (TIM1 and TIM8) consist of a 32-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

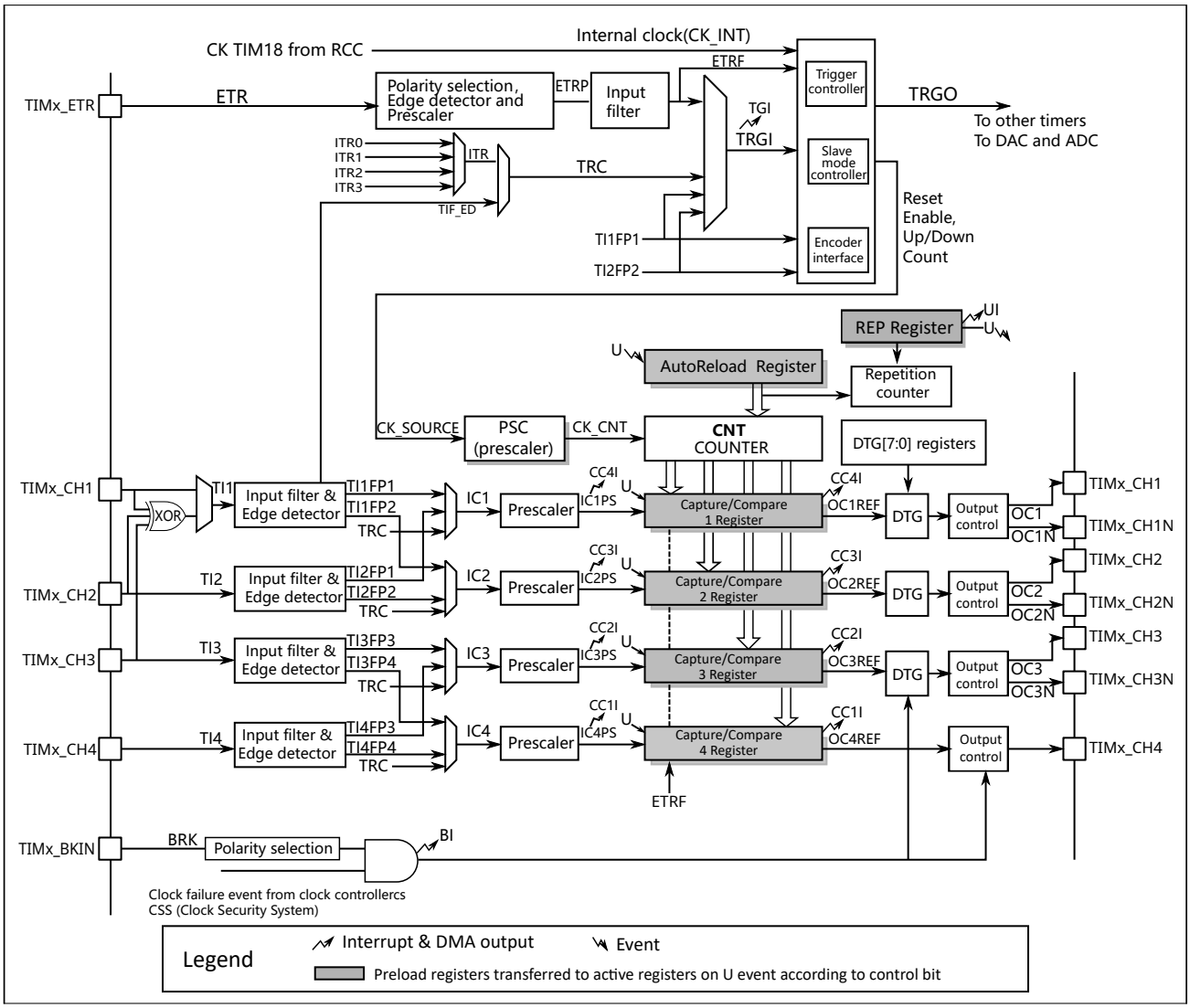
The advanced-control (TIM1 and TIM8) and general-purpose (TIMx) timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 15.3.15.

14.2 TIM1 and TIM8 main features

TIM1 and TIM8 timer features include:

- 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input Capture
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- Break input to put the timer’ s output signals in reset state or in a known state.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Fig 14.2-1 Advanced-control timer block diagram



14.3 TIM1 and TIM8 functional description

14.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 32-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: *that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.*

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 53 and Figure 54 give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Fig 14.3-1 Counter timing diagram with prescaler division change from 1 to 2

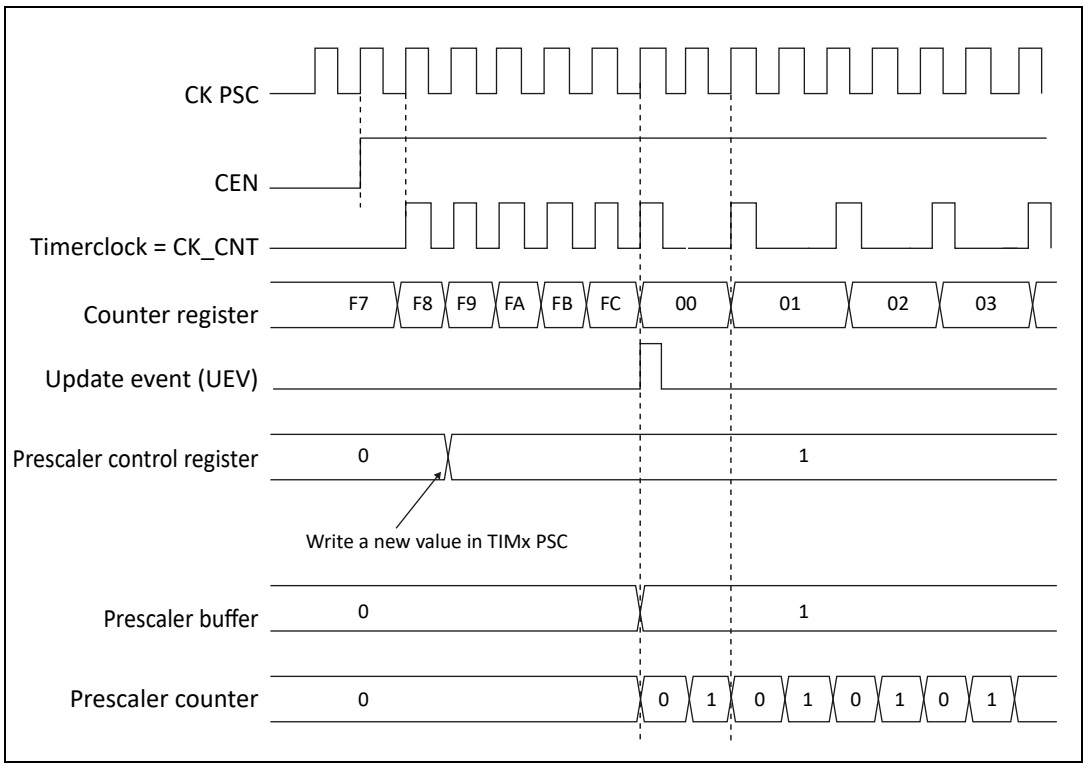
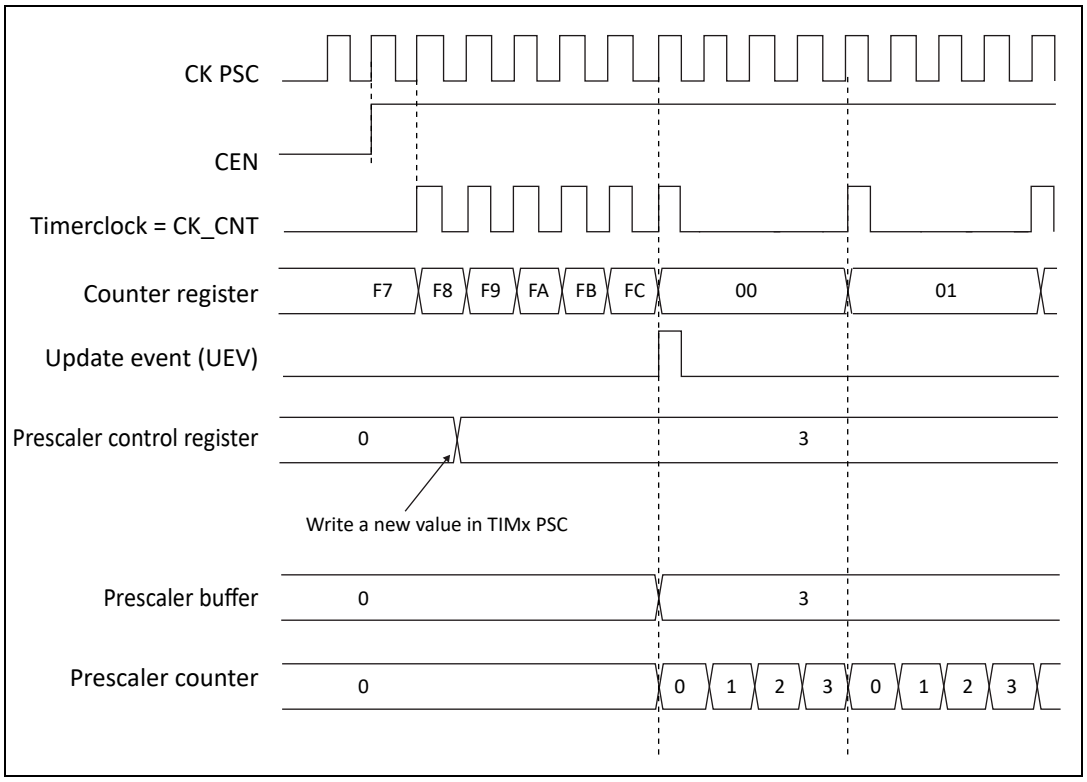


Fig 14.3-2 Counter timing diagram with prescaler division change from 1 to 4



14.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register plus one (TIMx_RCR+1). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,

- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

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

Fig 14.3-3 Counter timing diagram, internal clock divided by 1

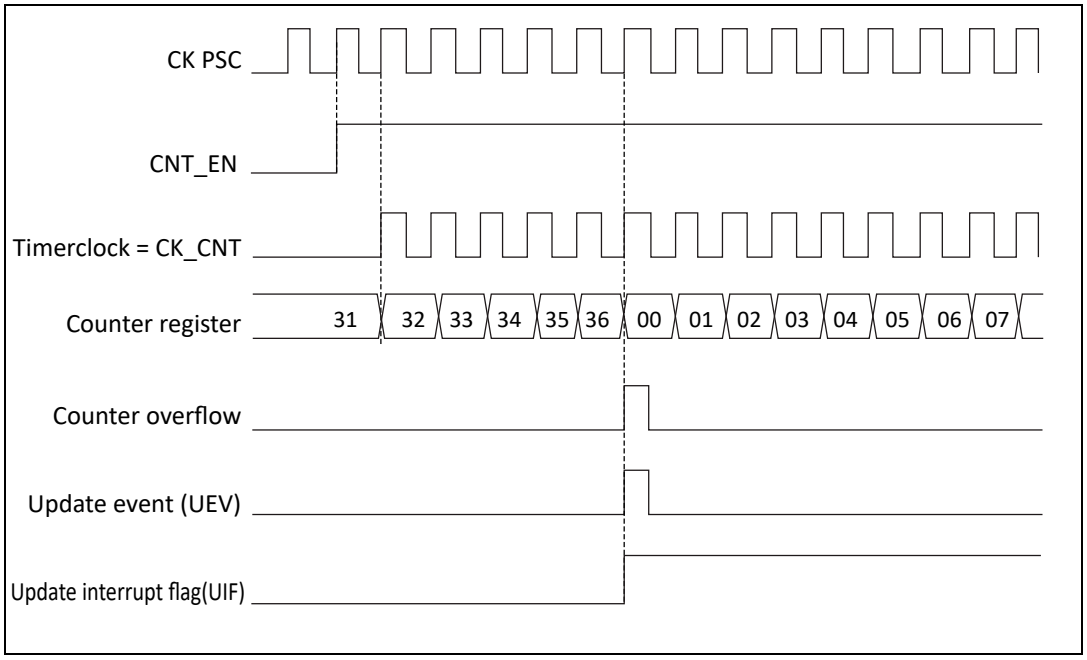


Fig 14.3-4 Counter timing diagram, internal clock divided by 2

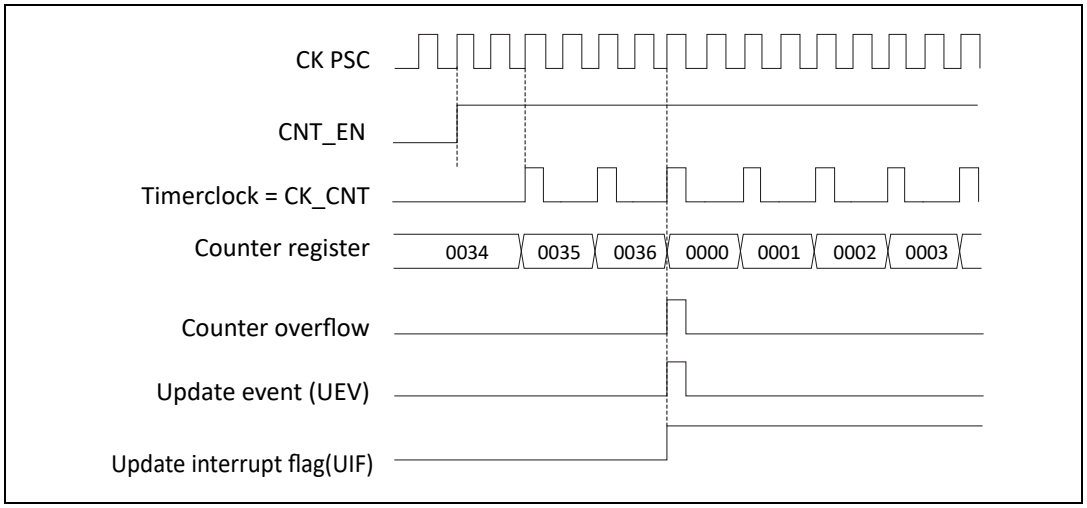


Fig 14.3-5 Counter timing diagram, internal clock divided by 4

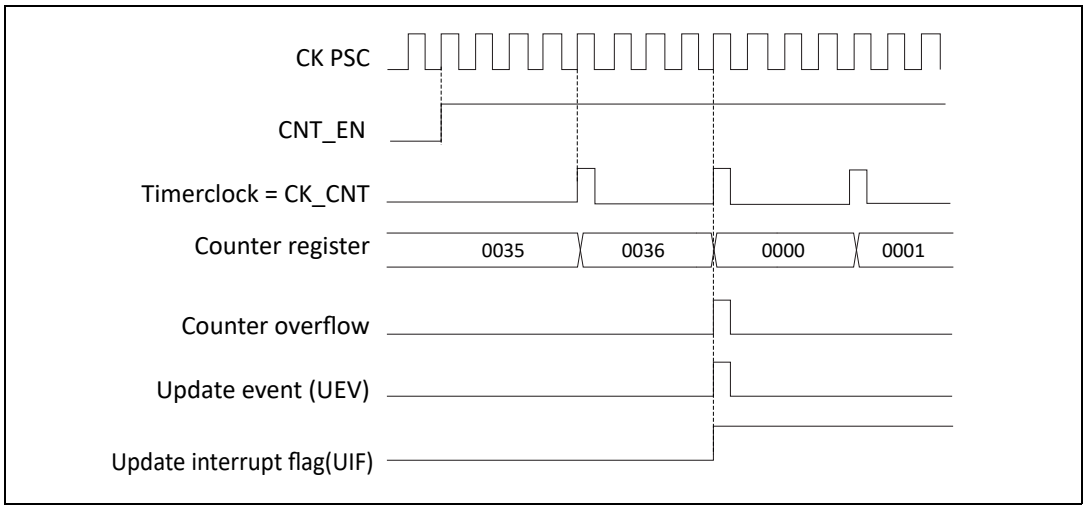


Fig 14.3-6 Counter timing diagram, internal clock divided by n

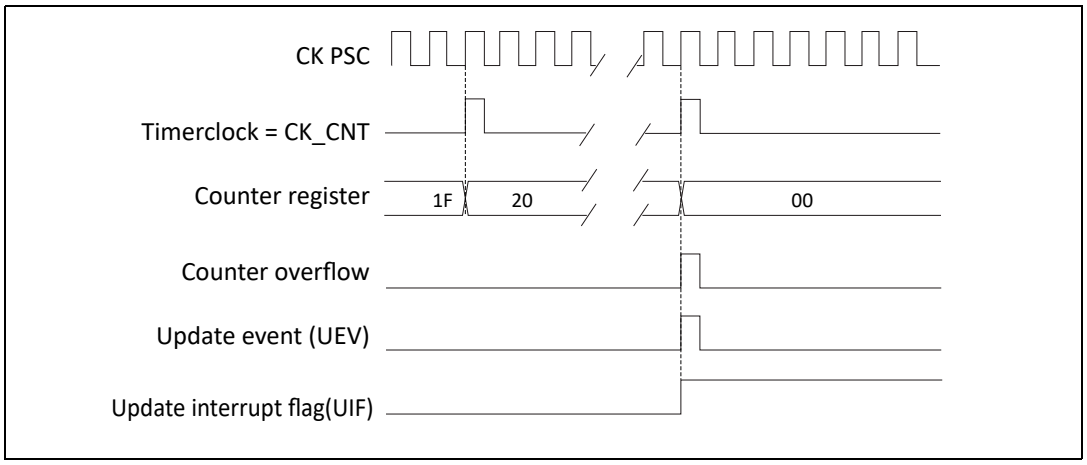


Fig 14.3-7 Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

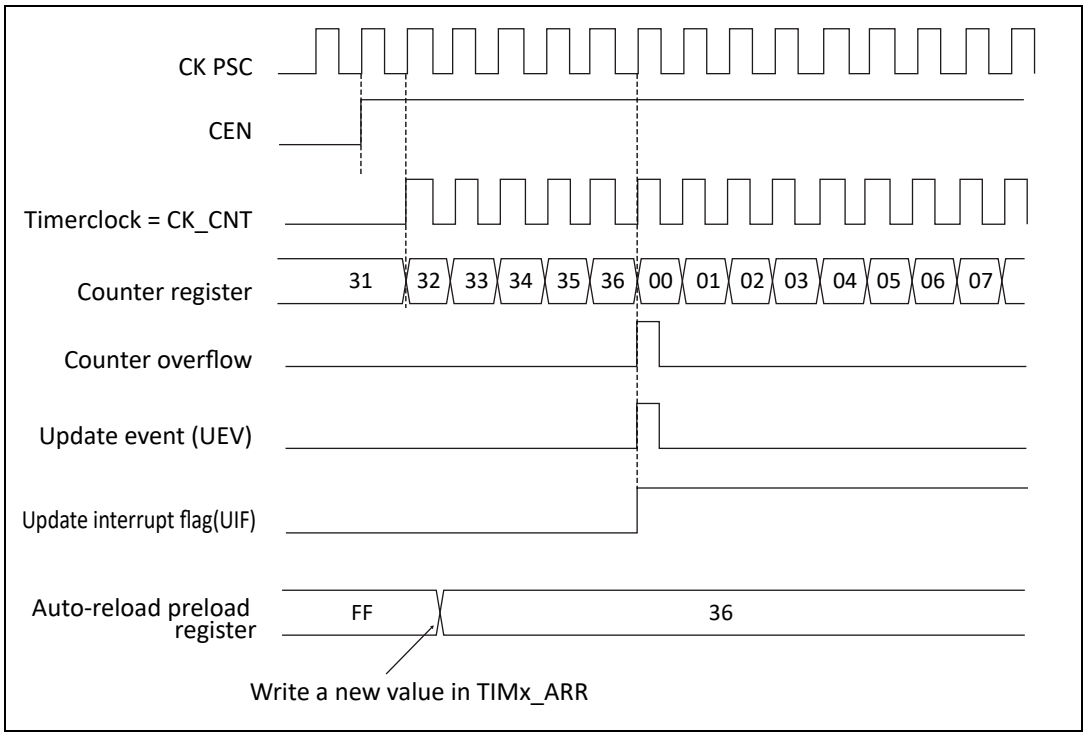
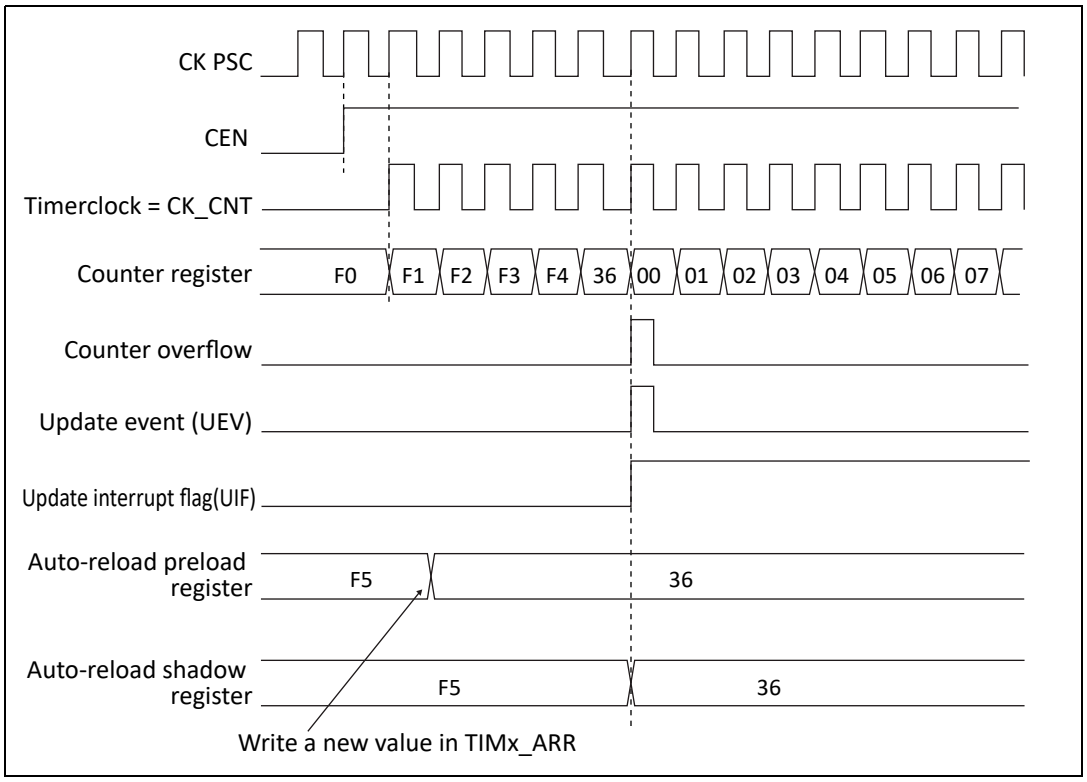


Fig 14.3-8 Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register plus one (TIMx_RCR+1). Else the update event is generated at each counter underflow. Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one

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

Fig 14.3-9 Counter timing diagram, internal clock divided by 1

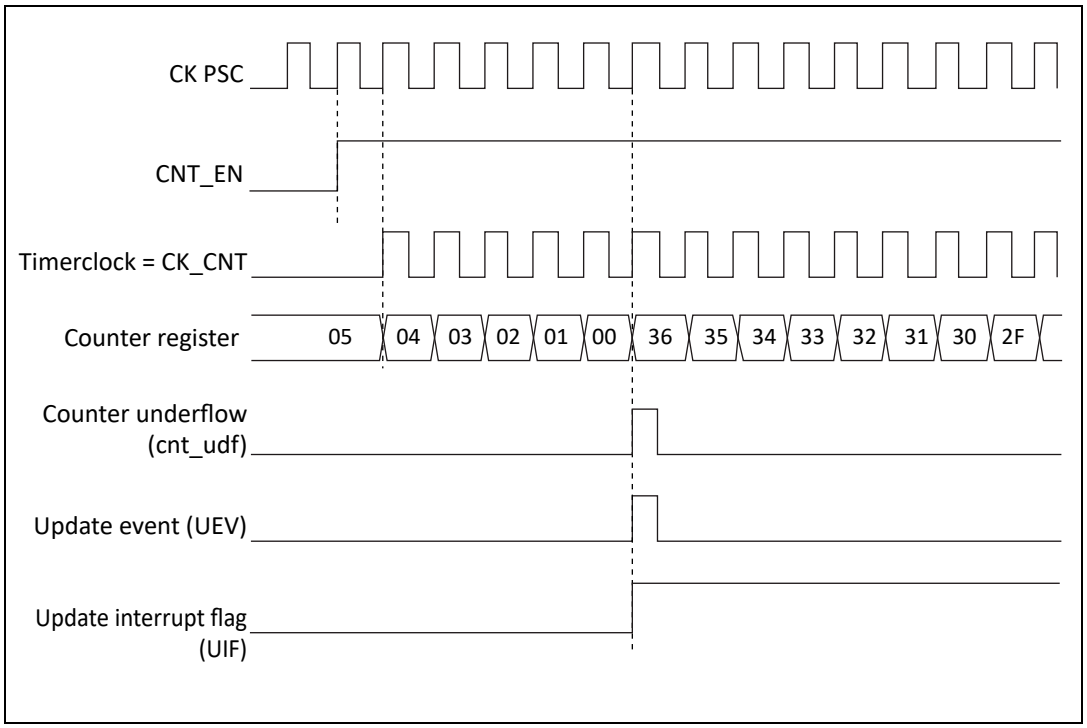


Fig 14.3-10 Counter timing diagram, internal clock divided by 2

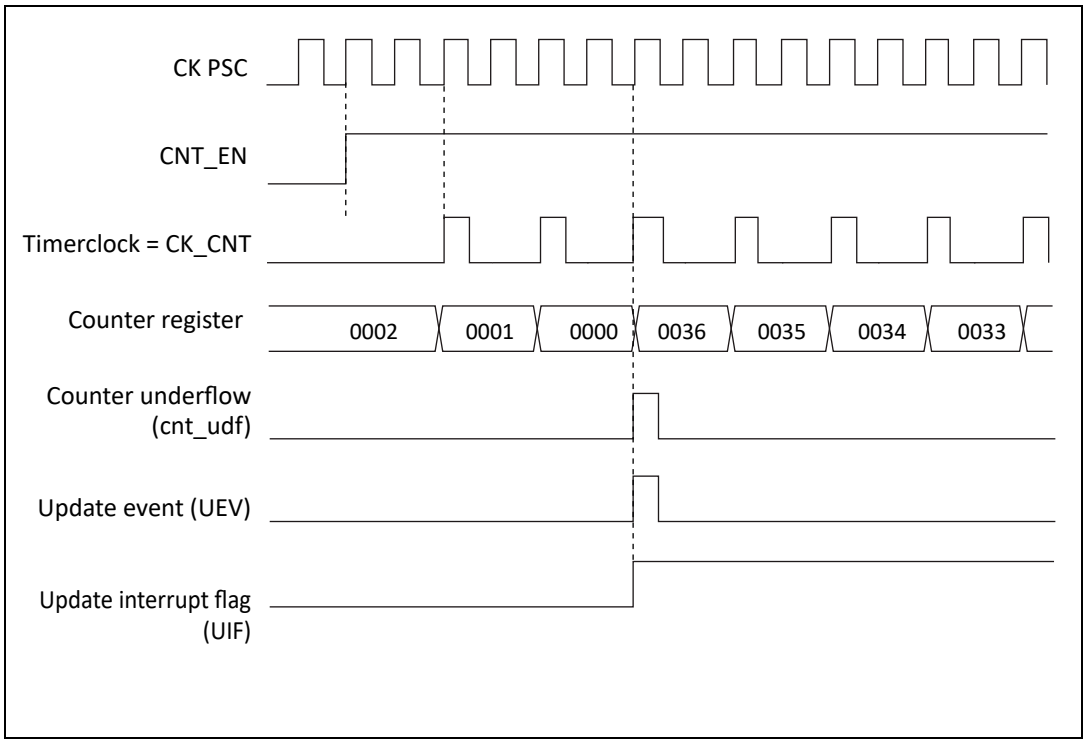


Fig 14.3-11 Counter timing diagram, internal clock divided by 4

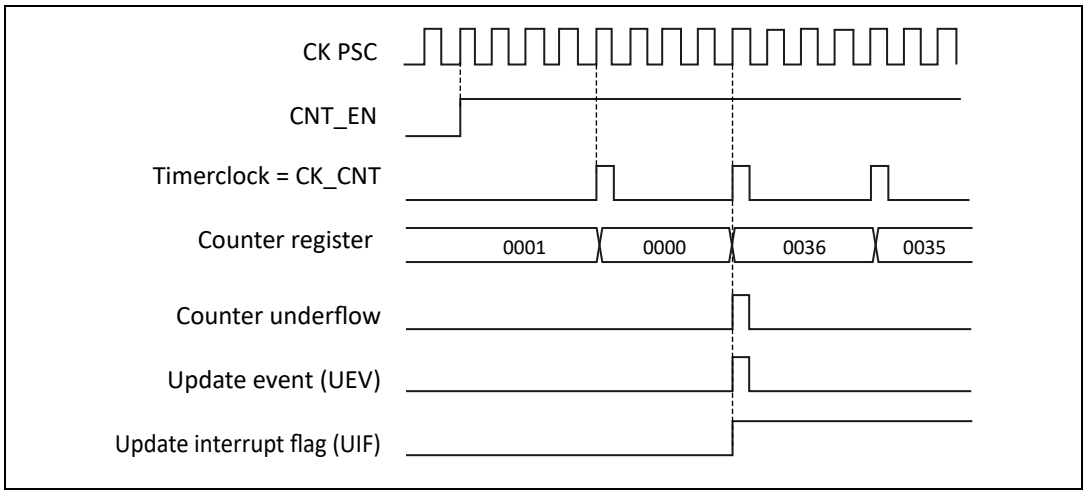


Fig 14.3-12 Counter timing diagram, internal clock divided by N

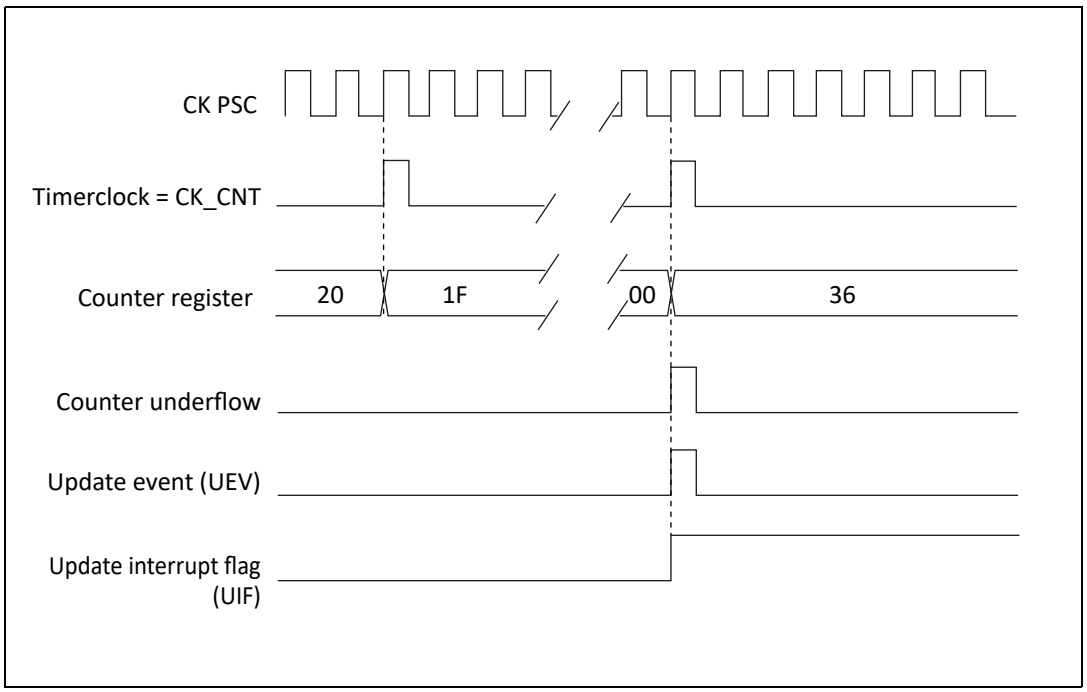
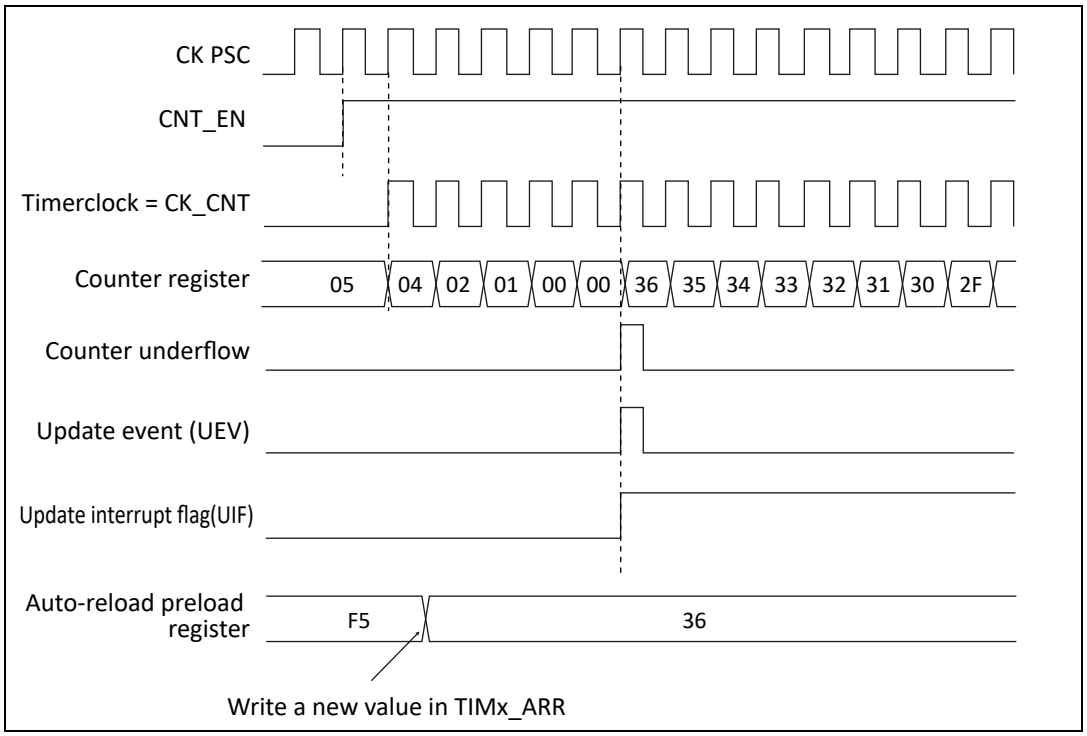


Fig 14.3-13 Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) –1, generates a counter overflow event, then counts from the auto- reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is

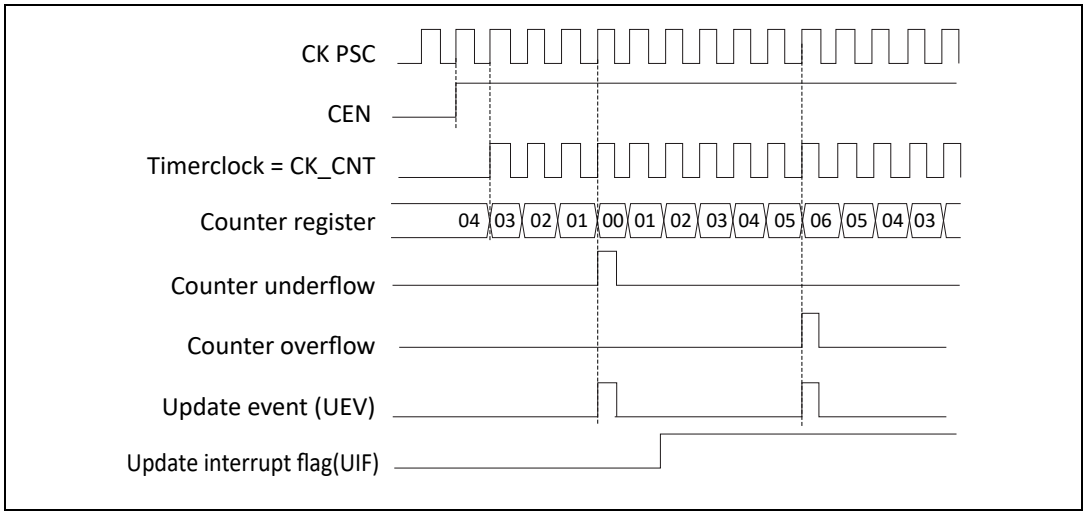
sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto- reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

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

Fig 14.3-14 Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used .

Fig 14.3-15 Counter timing diagram, internal clock divided by 2

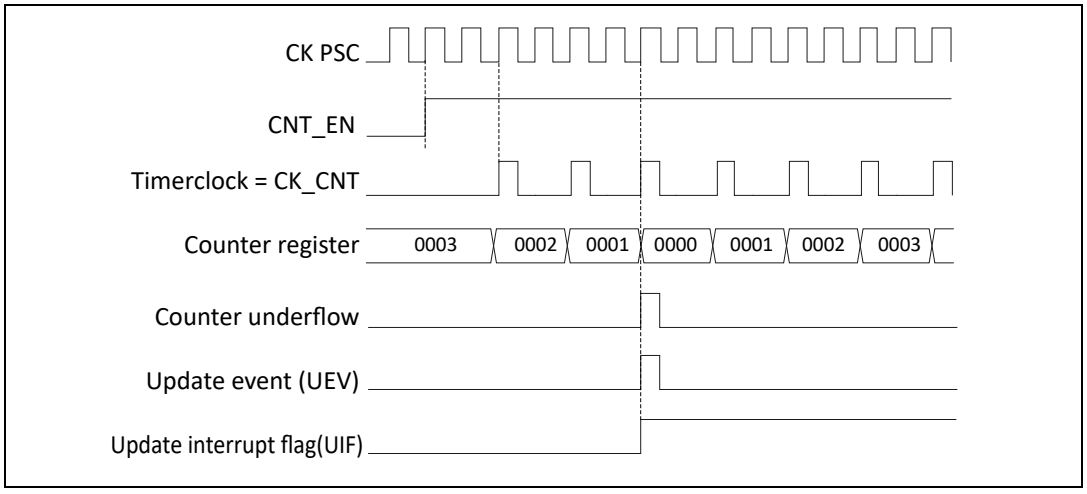
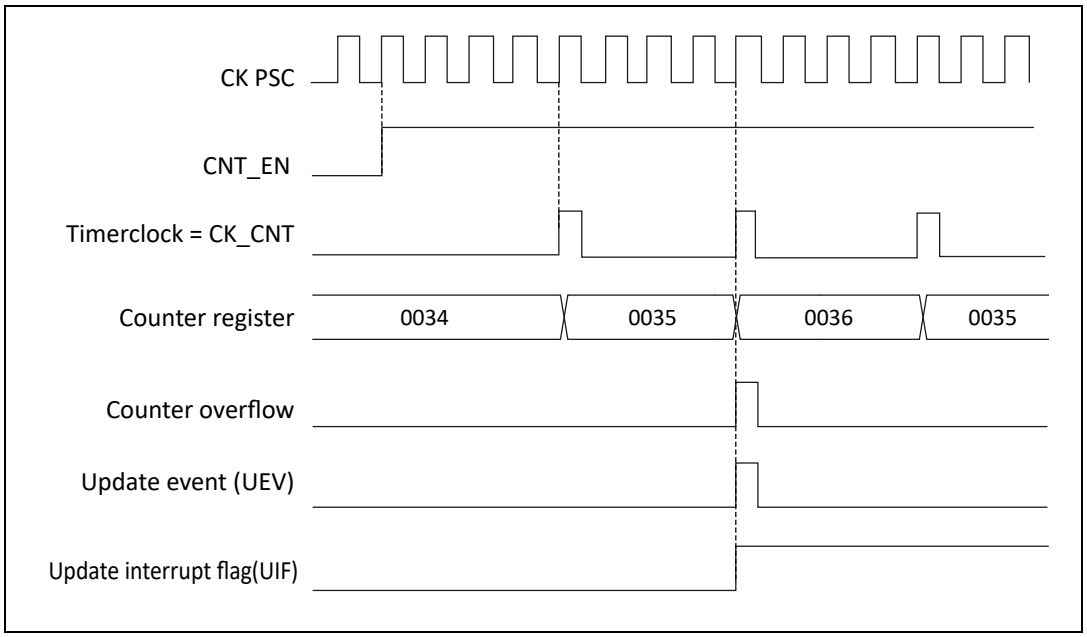


Fig 14.3-16 Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Fig 14.3-17 Counter timing diagram, internal clock divided by N

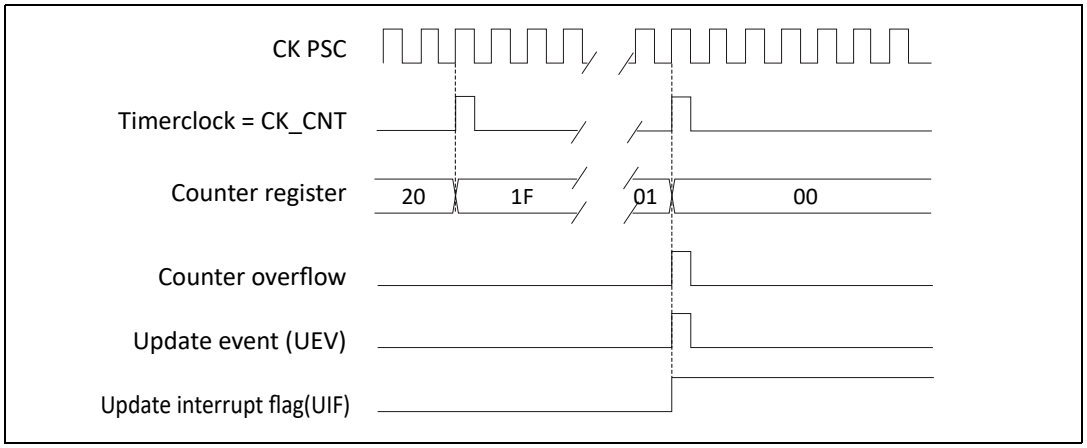


Fig 14.3-18 Counter timing diagram, update event with ARPE=1 (counter underflow)

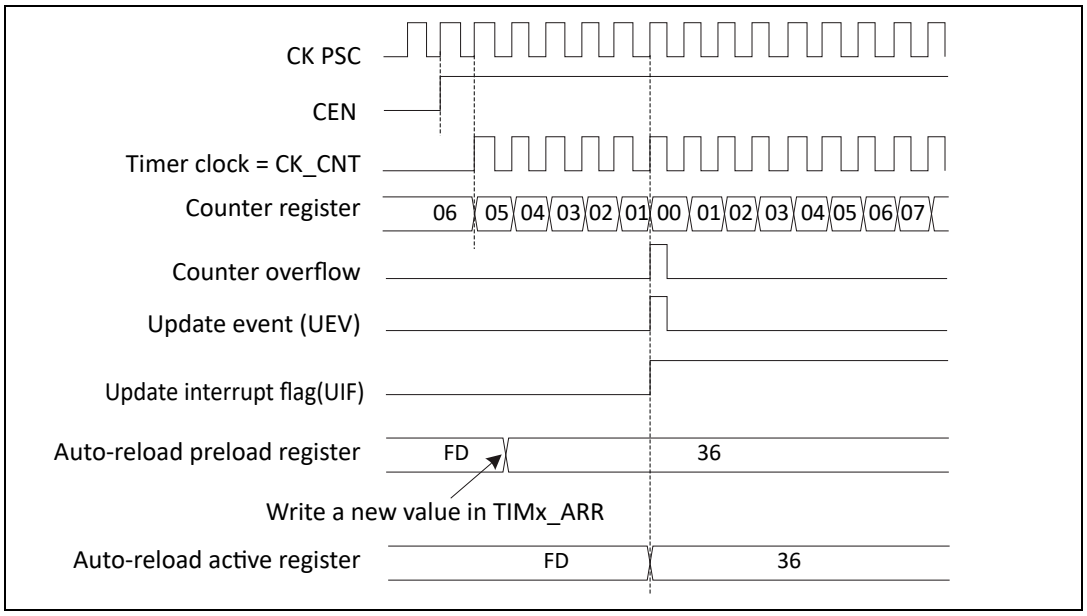
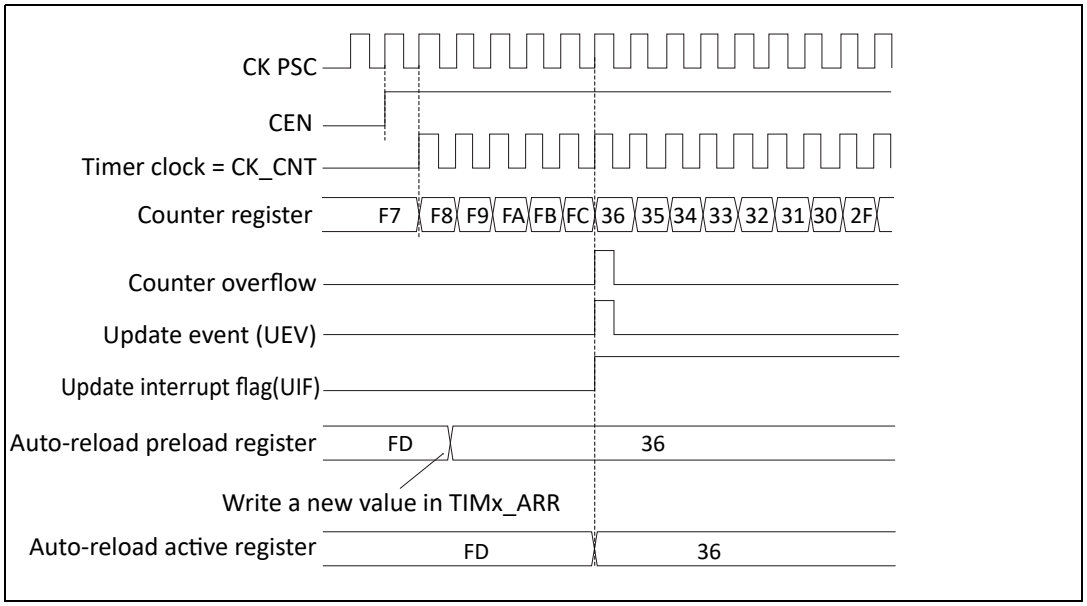


Fig 14.3-19 Counter timing diagram, Update event with ARPE=1 (counter overflow)



14.3.3 Repetition counter

Section 14.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

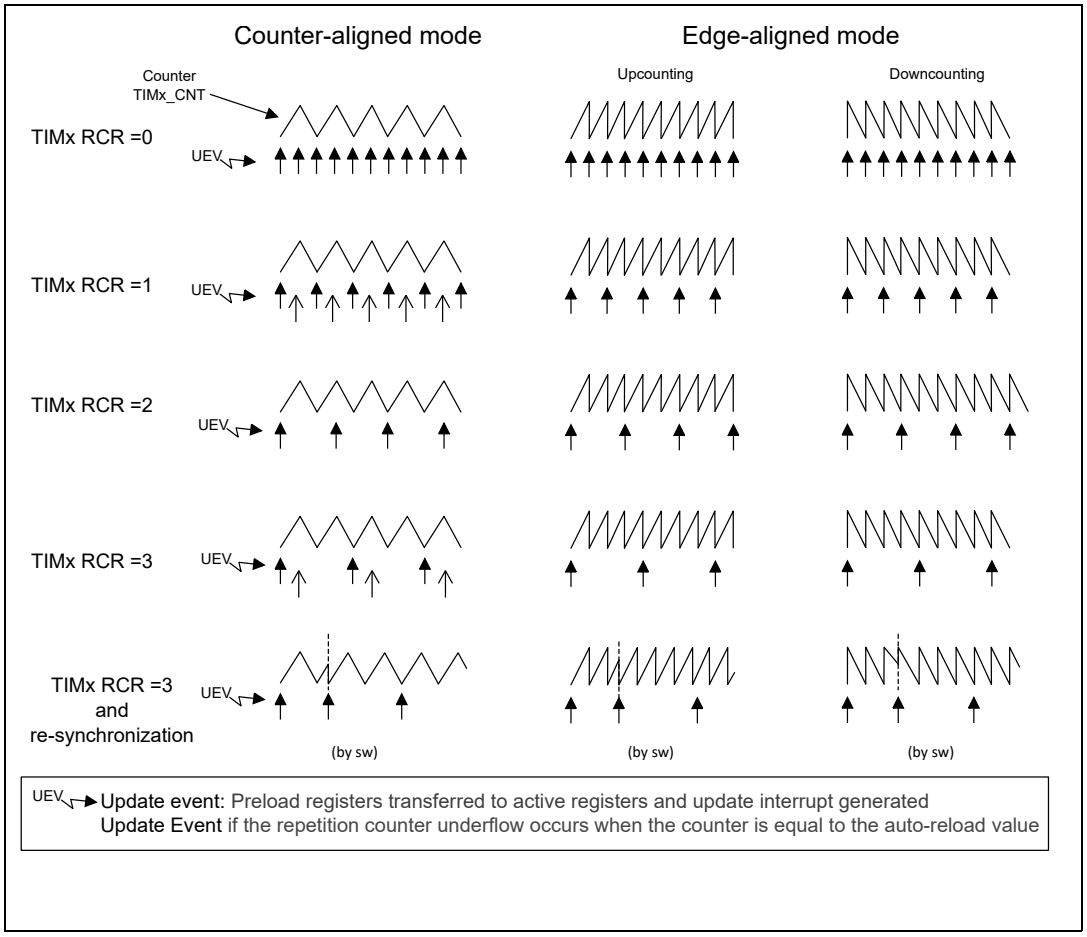
The repetition counter is decremented:

- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 128 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to Figure 72). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In center-aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was started. If the RCR was written before starting the counter, the UEV occurs on the overflow. If the RCR was written after starting the counter, the UEV occurs on the underflow. For example for RCR = 3, the UEV is generated on each 4th overflow or underflow event depending on when RCR was written.

Fig 14.3-20 Update rate examples depending on mode and TIMx_RCR register settings



14.3.4 Clock selection

The counter clock can be provided by the following clock sources:

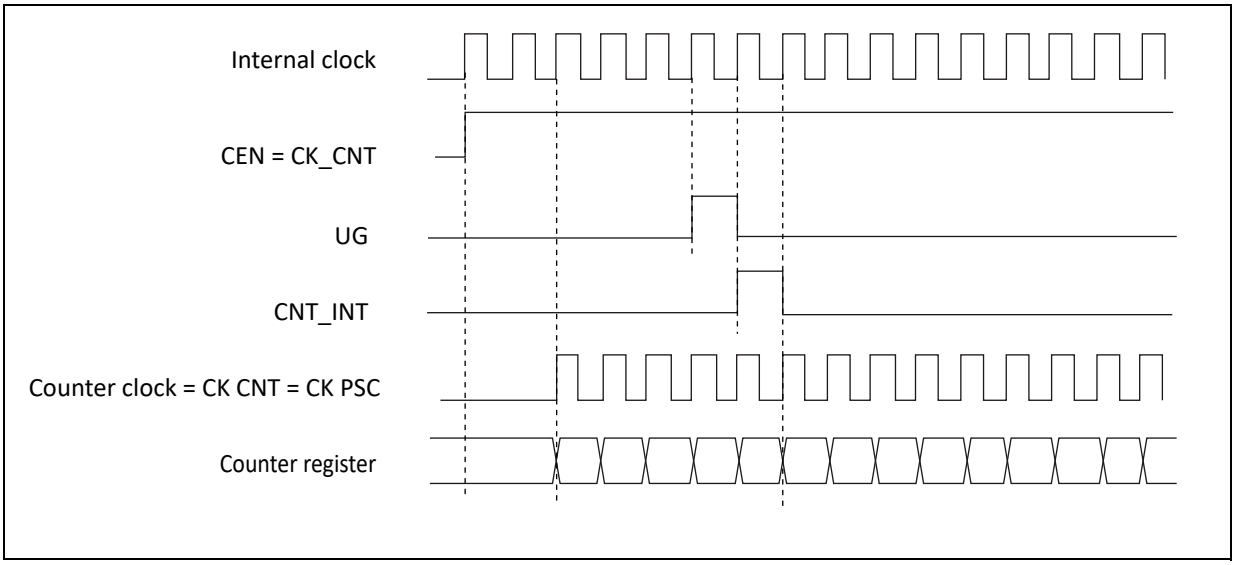
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, the user can configure Timer 1 to act as a prescaler for Timer 2.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 14.3-21 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

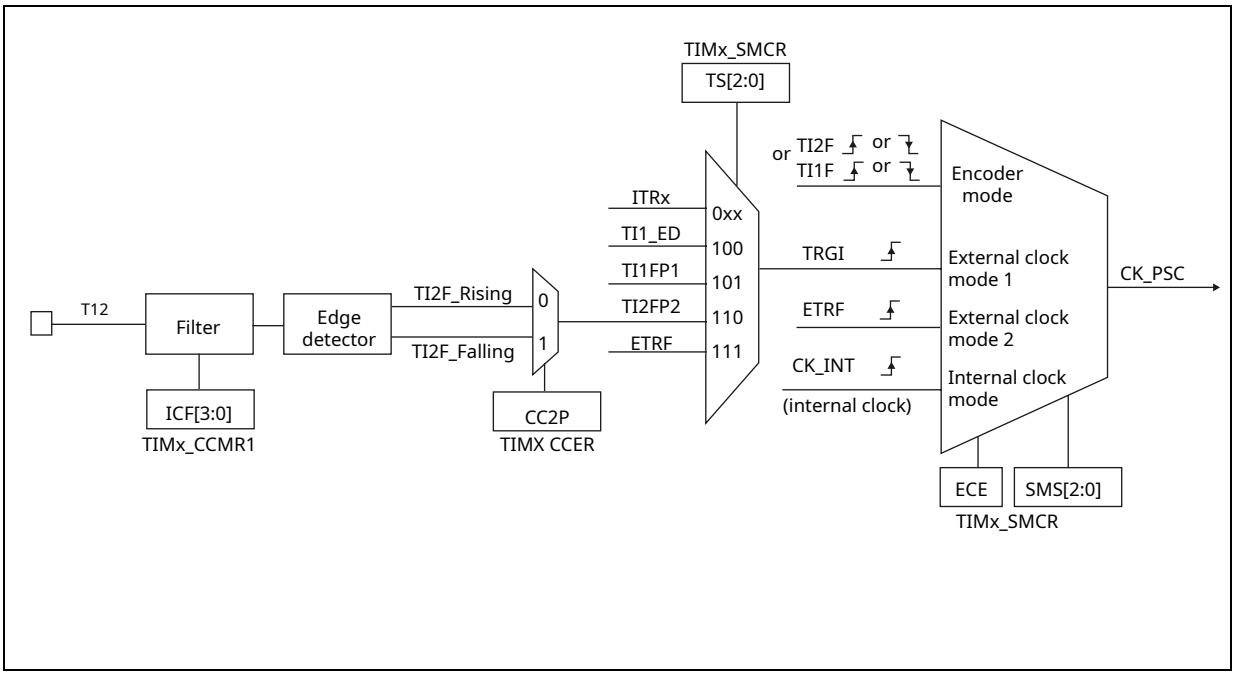
Fig 14.3-21 Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Fig 14.3-22 TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

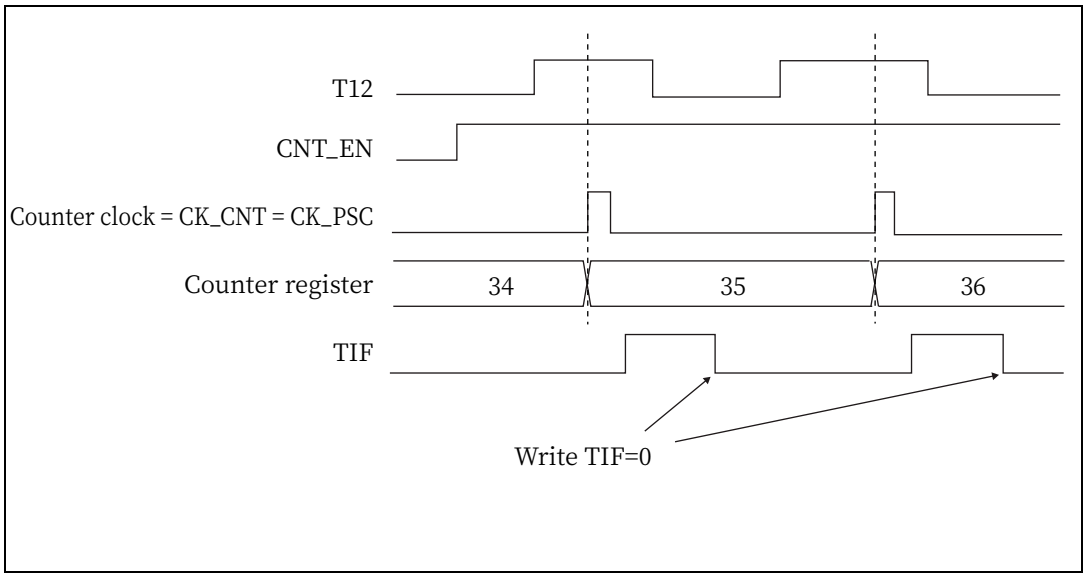
Note:

The capture prescaler is not used for triggering, so the user does not need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Fig 14.3-23 Control circuit in external clock mode 1

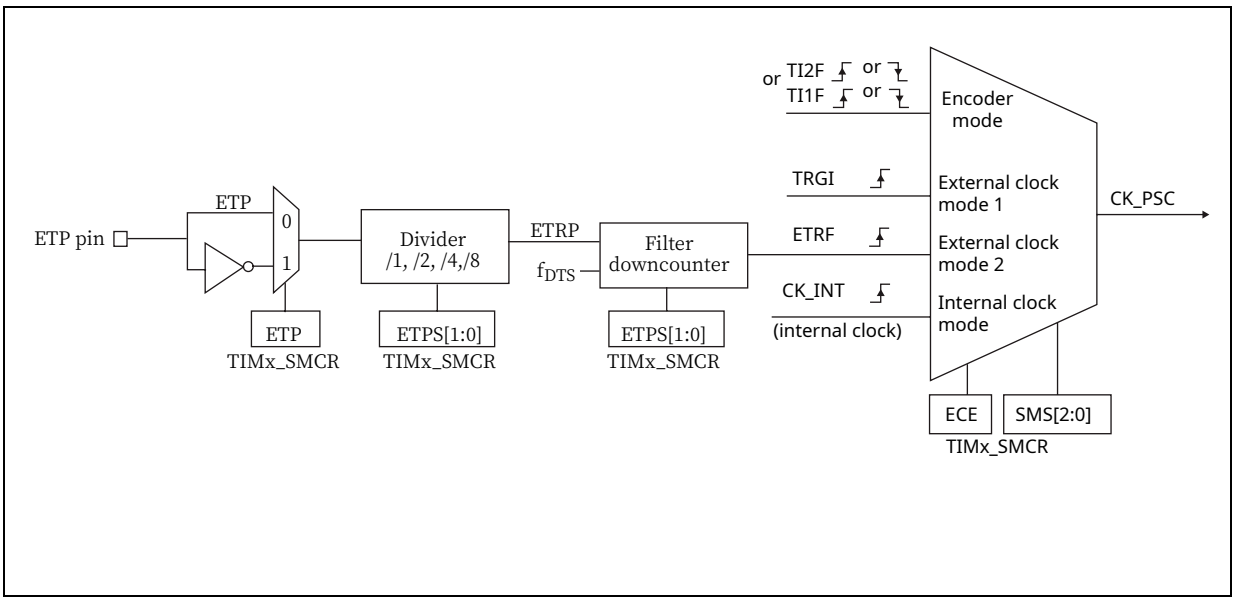


External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

Fig 14.3-24 External trigger input block



For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

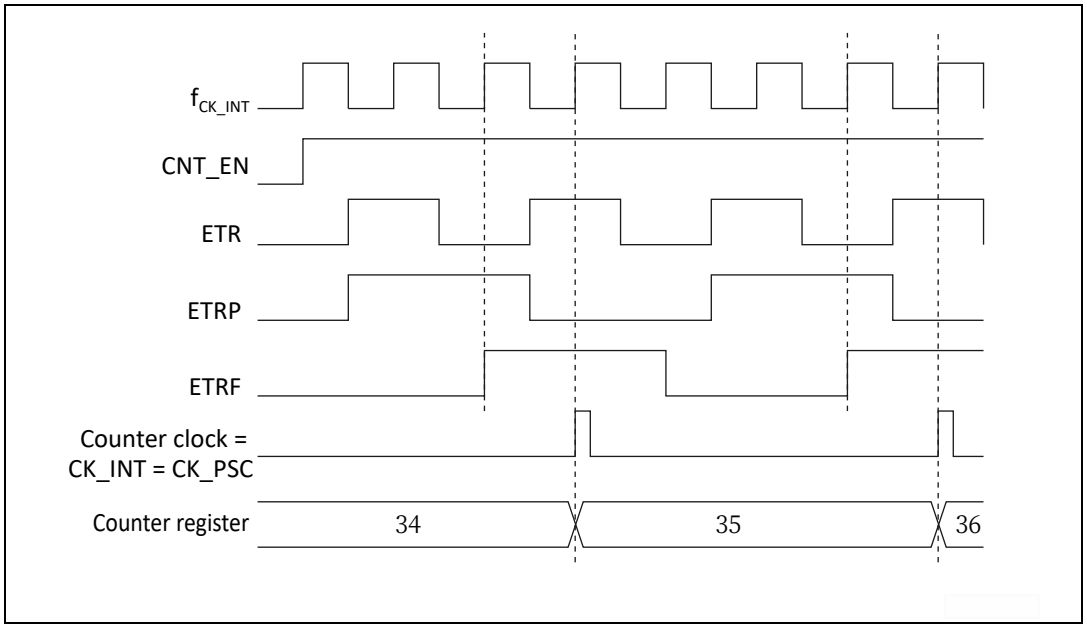
1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register

4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Fig 14.3-25 Control circuit in external clock mode 2

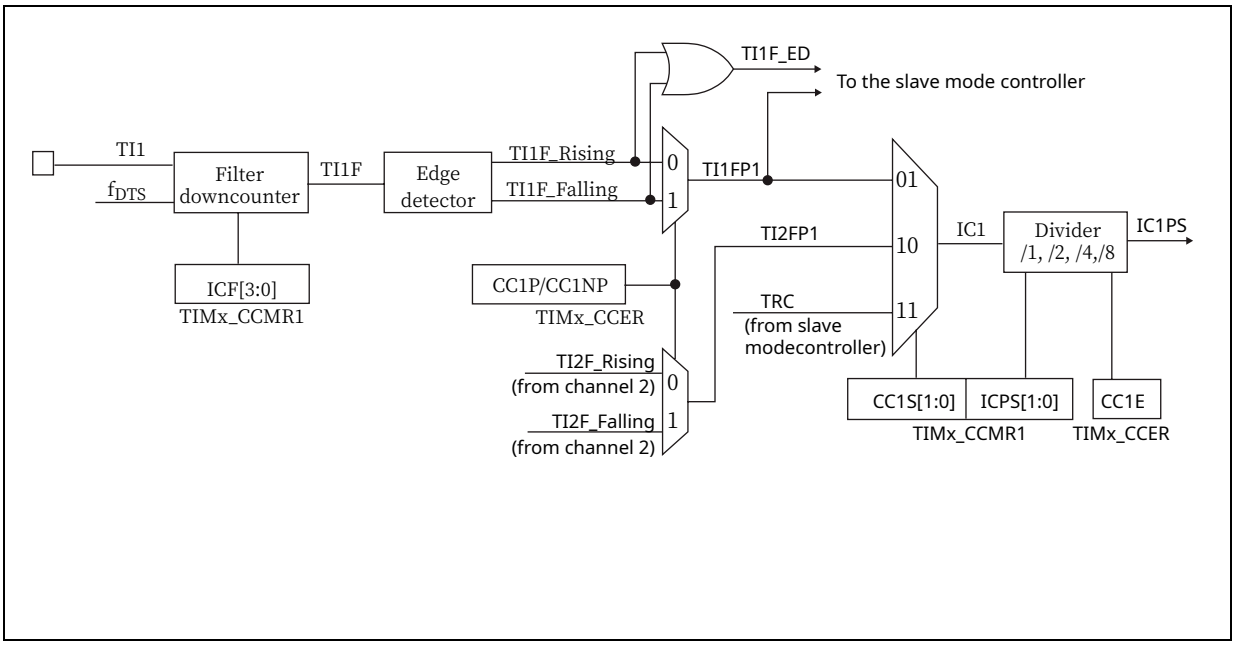


14.3.5 Capture/compare channels

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).

Fig 14.3-26 Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform that is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Fig 14.3-27 Capture/compare channel 1 main circuit

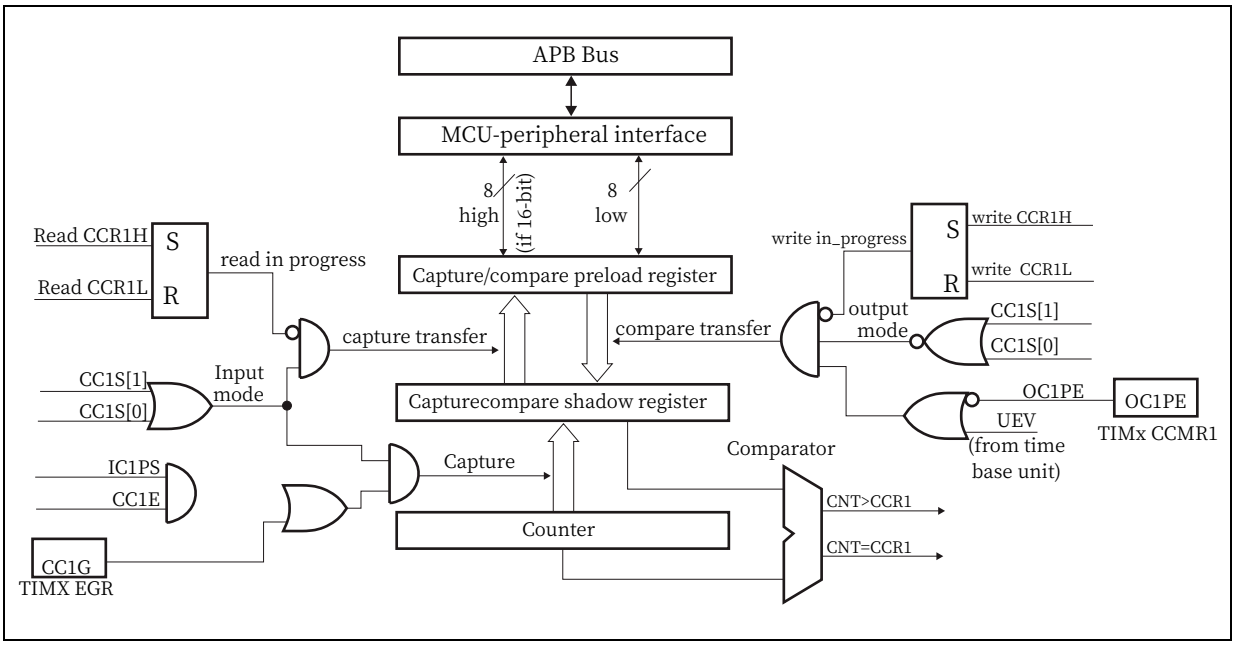


Fig 14.3-28 Output stage of capture/compare channel (channel 1 to 3)

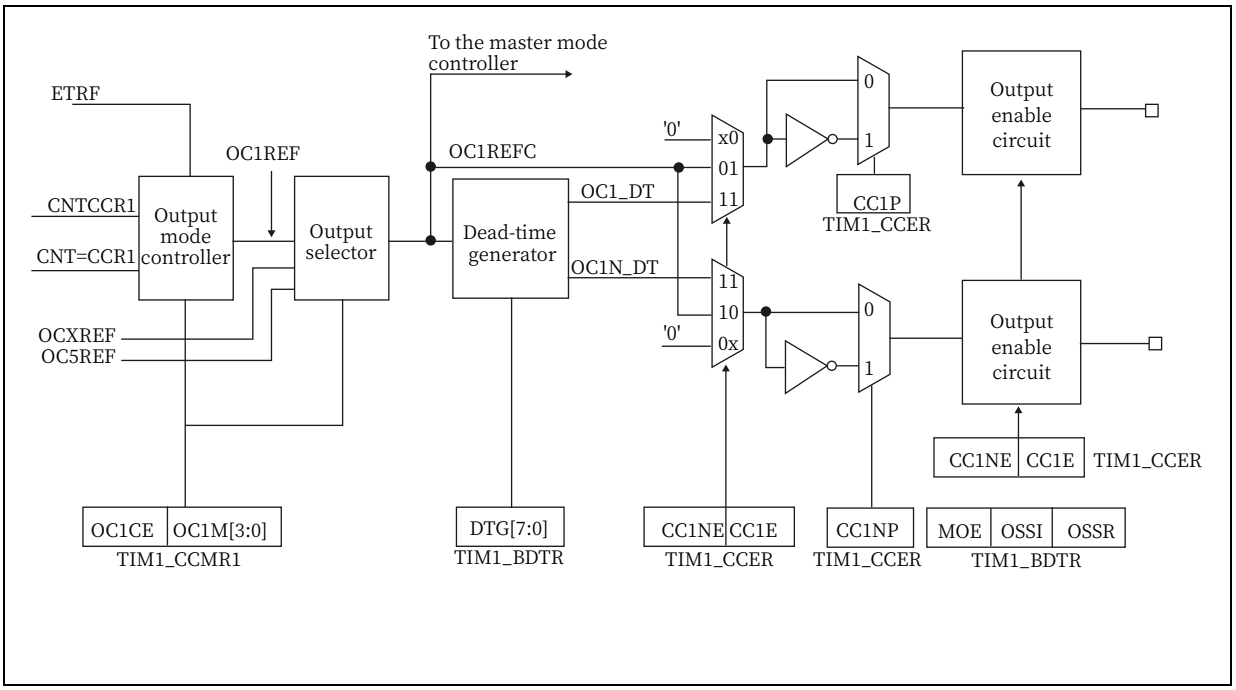
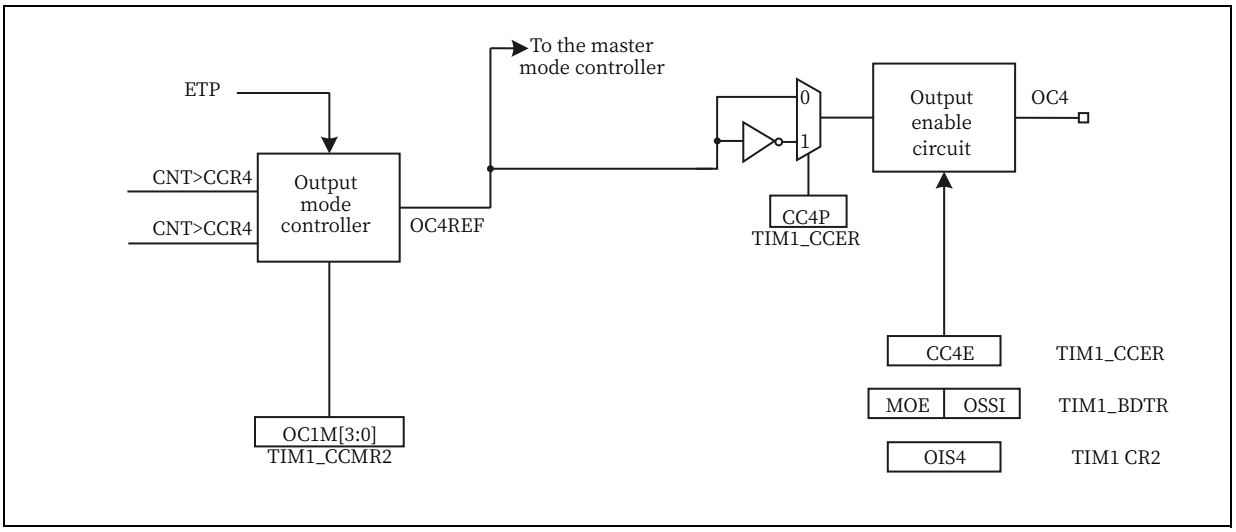


Fig 14.3-29 Output stage of capture/compare channel (channel 4)



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.

14.3.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written to ‘0’.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- 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 and the TIMx_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the signal connected to the timer (by programming ICxF bits in the TIMx_CCMRx register if the input is a TIx input). Let’s imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f DTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
- 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, and/or the DMA request by setting the CC1DE 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.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

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 and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.*

14.3.7 PWM input mode

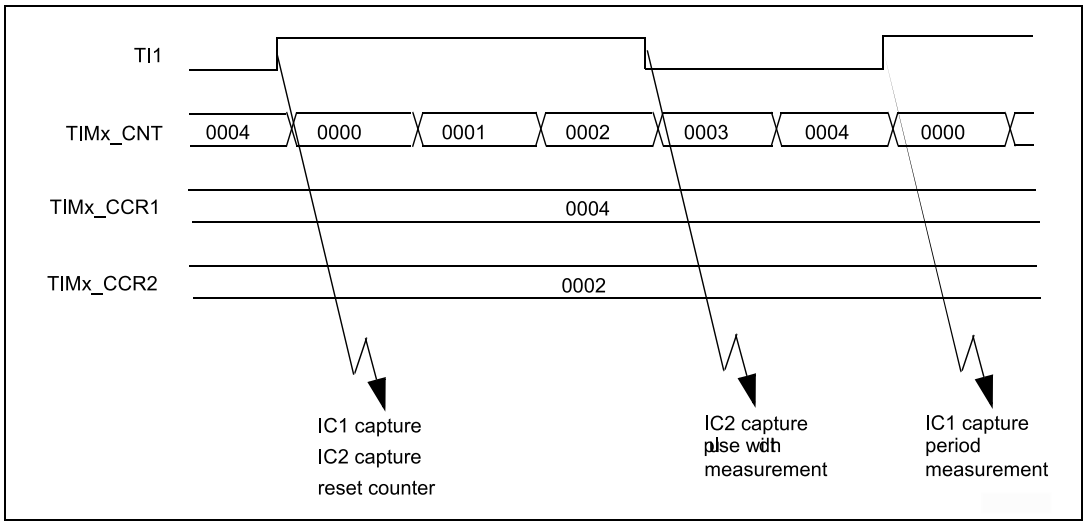
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P bit to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Fig 14.3-30 Output stage of capture/compare channel (channel 4)



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

14.3.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of

any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

14.3.9 Output compare mode

This function is used to control an output waveform or indicating 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).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

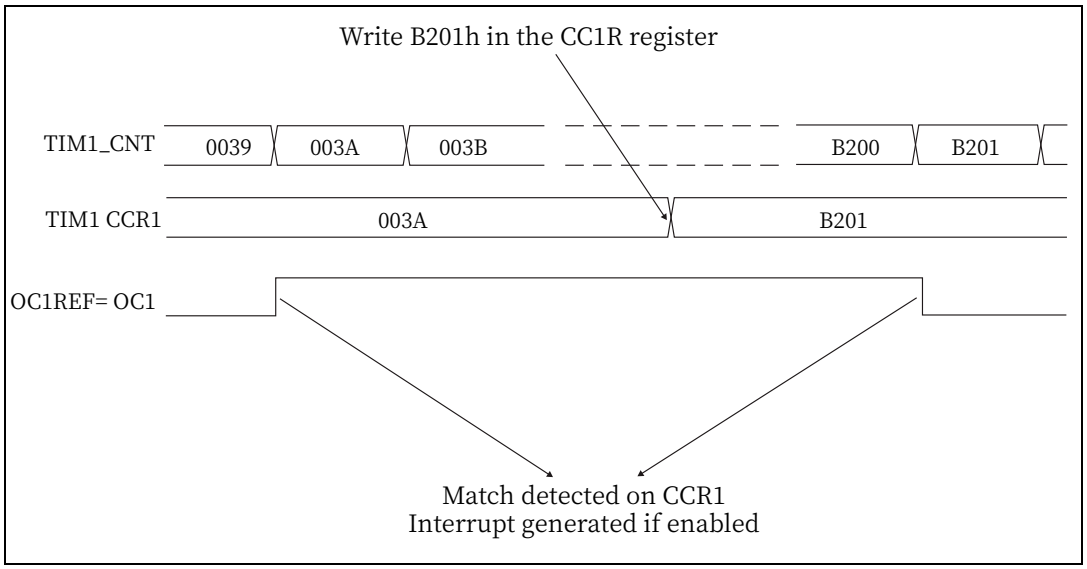
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
    - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
    - Write OCxPE = 0 to disable preload register
    - Write CCxP = 0 to select active high polarity
    - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=’0’, else TIMx_CCRx shadow register is updated only at the next update event UEV)

Fig 14.3-31 Output compare mode, toggle on OC1.



14.3.10 PWM mode

Pulse Width Modulation mode allows generating a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’(PWM mode 1) or ‘111’(PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user must initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

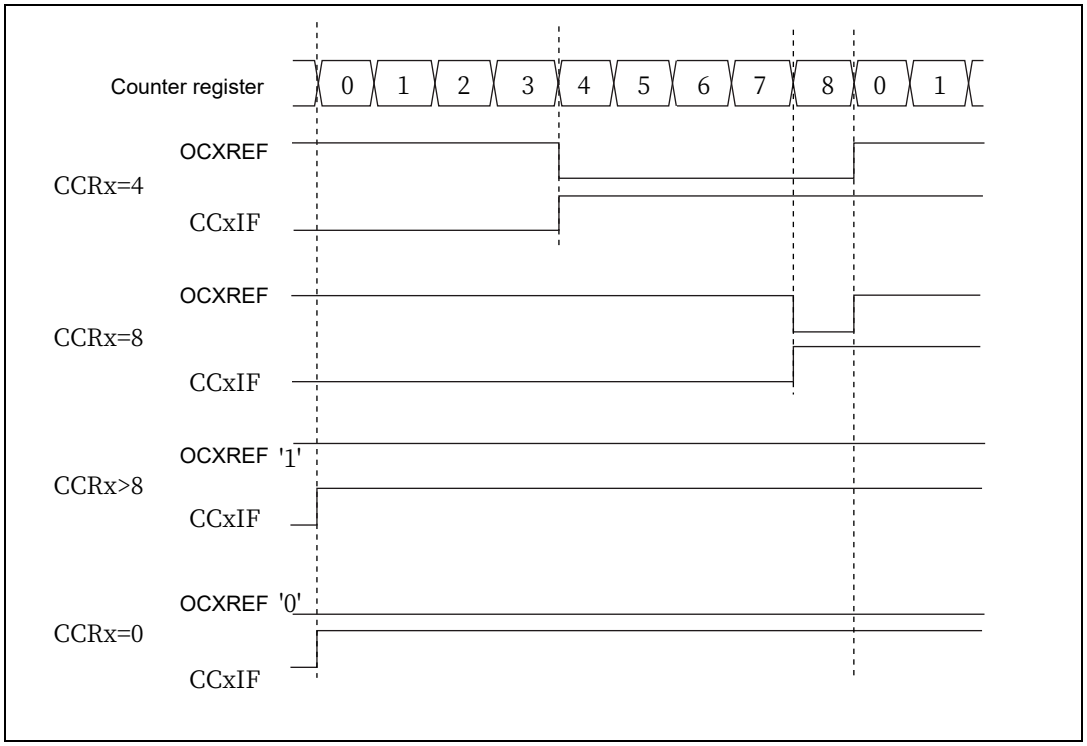
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx $\leqslant$ TIMx_CNT or TIMx_CNT $\leqslant$ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

- Upcounting configuration
  Upcounting is active when the DIR bit in the TIMx_CR1 register is low.
  In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. Figure 14.3-32 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Fig 14.3-32 Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration
  Downcounting is active when DIR bit in TIMx_CR1 register is high.
  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

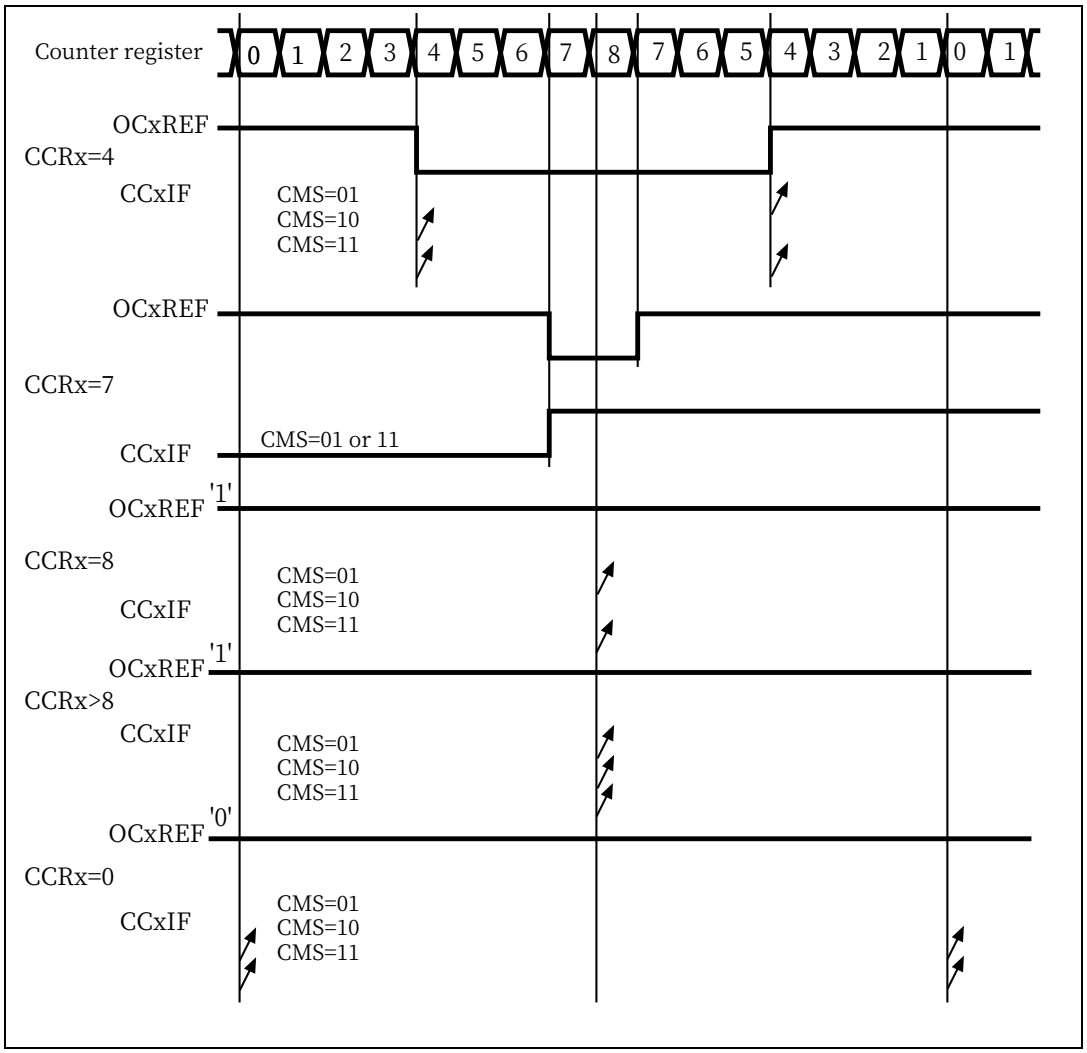
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’(all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software.

Figure 85 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Fig 14.3-33 Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if the user writes a value in the counter greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it will continue to count up.
  - The direction is updated if the user writes 0 or write the TIMx_ARR value in the counter but no Update Event UEV is generated.

- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

14.3.11 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1 and TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjust it depending on the devices connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

User can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 15.4-13 for more details. In particular, the dead-time is activated when switching to the IDLE state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. DTG[7:0] bits of the TIMx_BDTR register are used to control the dead-time generation for all channels. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Fig 14.3-34 Complementary output with dead-time insertion.

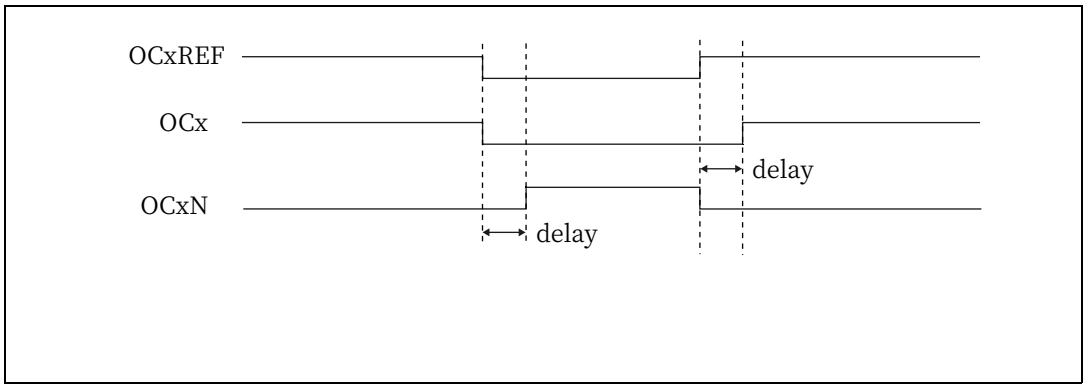


Fig 14.3-35 Dead-time waveforms with delay greater than the negative pulse

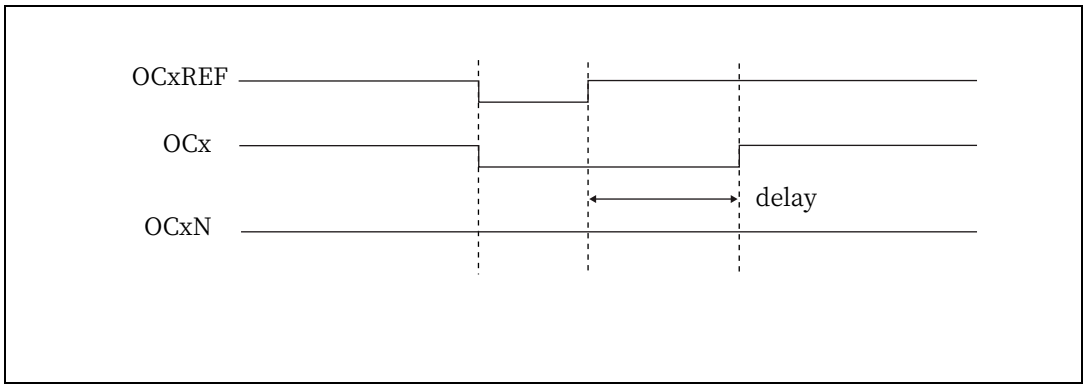
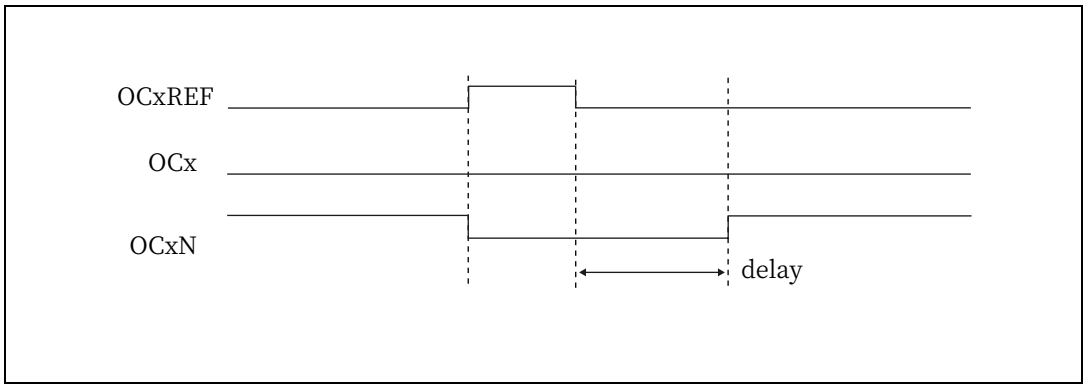


Fig 14.3-36 Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register.

Re-directing OCxREF to OCx or OCxN

n output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows the user to send a specific waveform (such as PWM or static active level) on one output

while the complementary remains at its inactive level. Other possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note : *When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.*

Using the break function

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to Table 15.4-13 for more details.

The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller. For further information on the Clock Security System.

When exiting from reset, the break circuit is disabled and the MOE bit is low. User can enable the break function by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is written to 1 whereas it was low, a delay (dummy instruction) must be inserted before reading it correctly. This is because the user writes an asynchronous signal, but reads a synchronous signal.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or in reset state (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0 then the timer releases the enable output else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).

  - If OSSI=0 then the timer releases the enable outputs else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

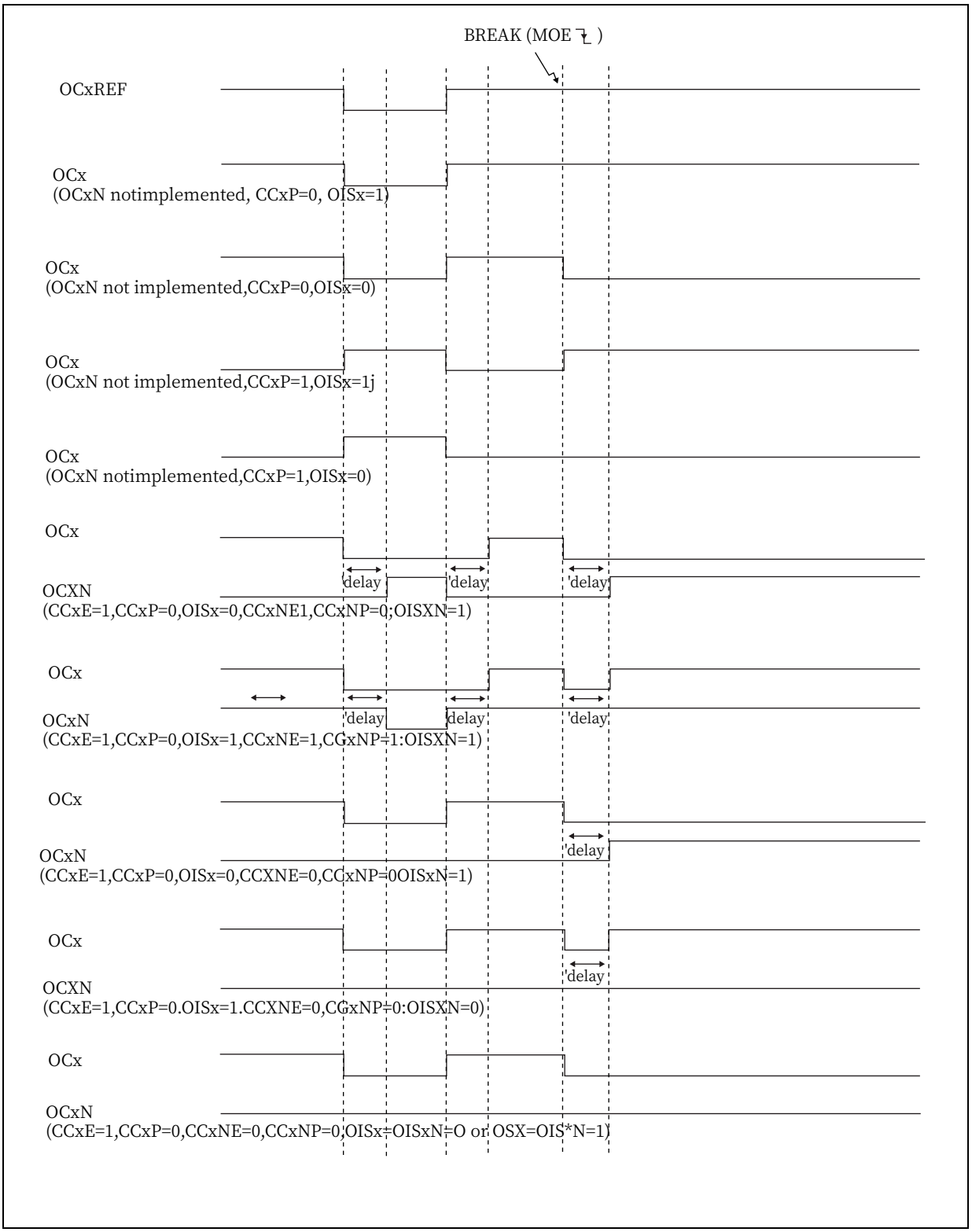
Note: *The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared. The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.*

There are two solutions to generate a break:

- By using the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register
- By software through the BG bit of the TIMx_EGR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows freezing the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The user can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to Section 14.4.18: TIM1 and TIM8 break and dead-time register (TIMx_BDTR). The LOCK bits can be written only once after an MCU reset.

Fig 14.3-37 Output behavior in response to a break



14.3.12 Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OCxCE enable bit of the corresponding TIMx_CCMRx register set to ‘1’). The OCxREF signal remains Low until the next update event, UEV, occurs.

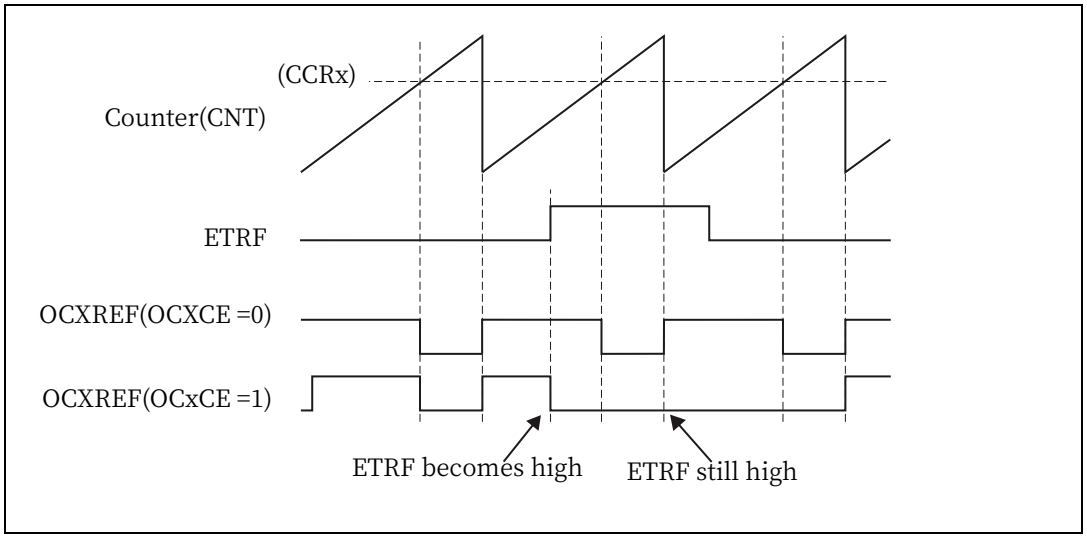
This function can only be used in output compare and PWM modes, and does not work in forced mode.

For example, the ETR signal can be connected to the output of a comparator to be used for current handling. In this case, the ETR must be configured as follow:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

Figure 14.3-38 shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Fig 14.3-38 Clearing TIMx OCxREF



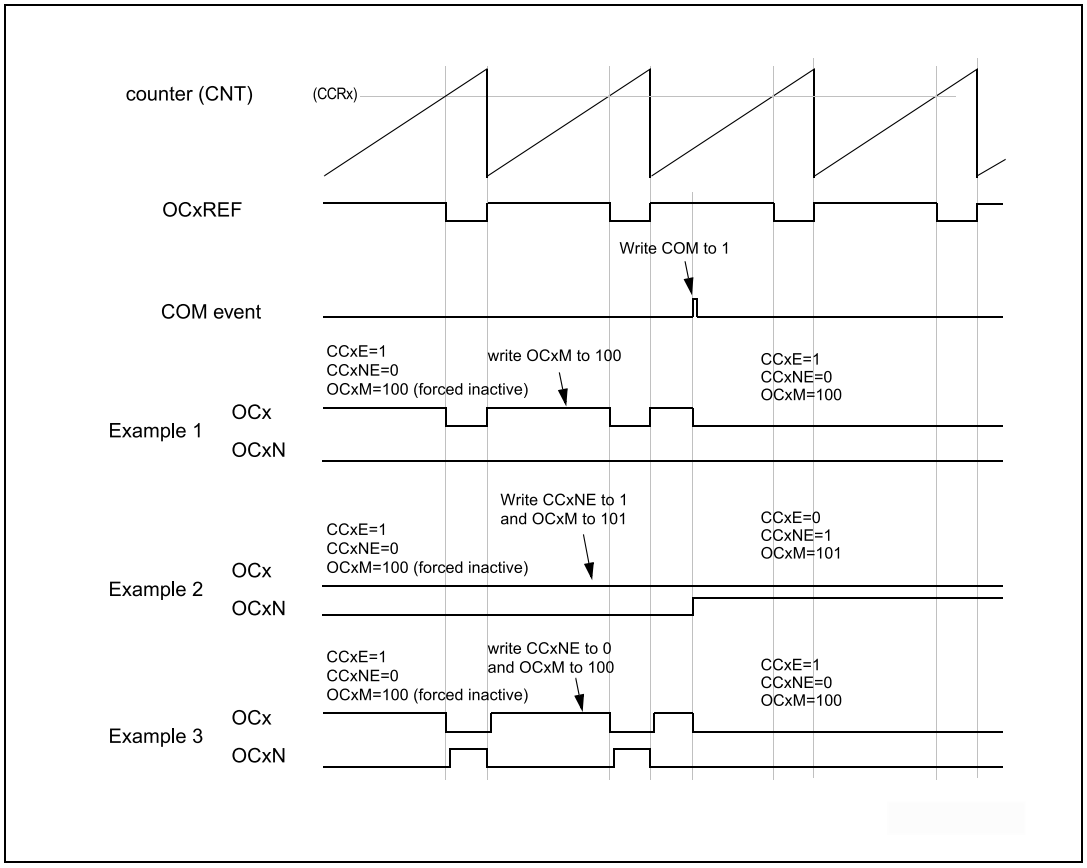
14.3.13 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. The user can thus program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

Figure 14.3-39 describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Fig 14.3-39 6-step generation, COM example (OSSR=1)



14.3.14 One-pulse mode

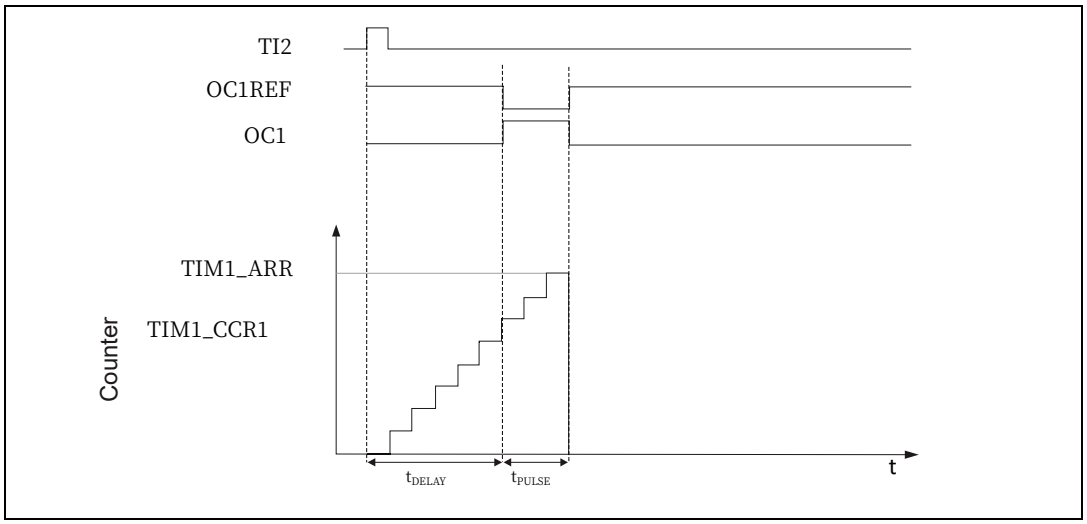
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx $\leqslant$ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Fig 14.3-40 Example of one pulse mode



For example the user may want to generate a positive pulse on OC1 with a length of t PULSE and after a delay of t DELAY as soon as a positive edge is detected on the TI2 input pin.

Let’s use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S=’01’ in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P=’0’ in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=’110’ in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to ‘110’ in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let us say the user wants to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t DELAY is defined by the value written in the TIMx_CCR1 register.
- The t PULSE is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let us say the user wants to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this, enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. The user can optionally enable the preload registers by writing OC1PE=’1’ in the TIMx_CCMR1 register and

ARPE in the TIMx_CR1 register. In this case the compare value must be written in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

The user only wants one pulse (Single mode), so ’1’ must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to ’0’, so the Repetitive Mode is selected.

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 t_{DELAY} min we can get.

If the user wants to output a waveform with the minimum delay, the OCxFE bit in the TIMx_CCMRx register must be set. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

14.3.15 Encoder interface mode

To select Encoder Interface mode write SMS=‘001’in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=’ 010’ if it is counting on TI1 edges only and SMS=’ 011’ if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the user can program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to Table 81. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1’). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So user must configure TIMx_ARR before starting. in the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder’s position. The count

direction correspond to the rotation direction of the connected sensor.

Table 14.3-1 summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Tab 14.3-1 Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Figure14.3-41 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S=’01’(TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S=’01’(TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P=’0’, and IC1F = ‘0000’(TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P=’0’, and IC2F = ‘0000’(TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS=’011’(TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN=’1’(TIMx_CR1 register, Counter enabled).

Fig 14.3-41 Example of counter operation in encoder interface mode

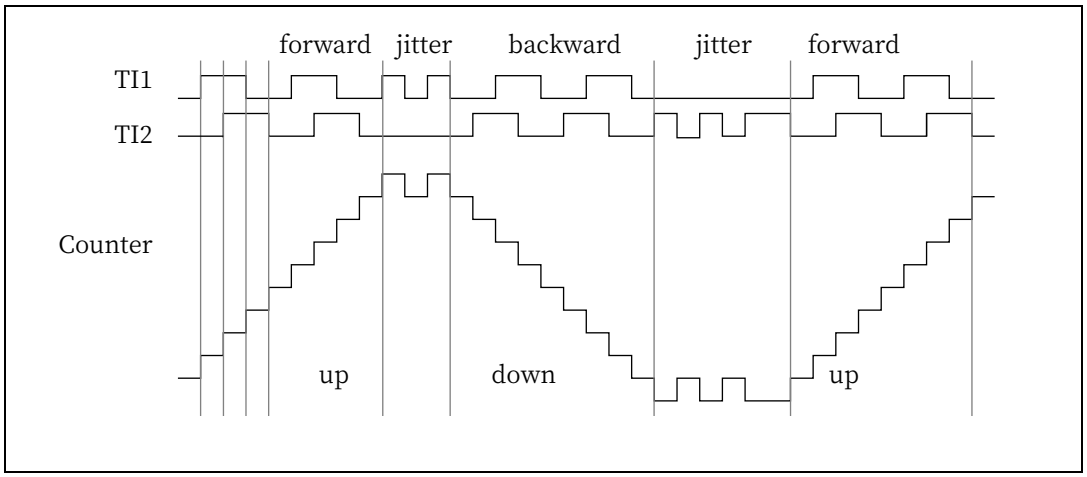
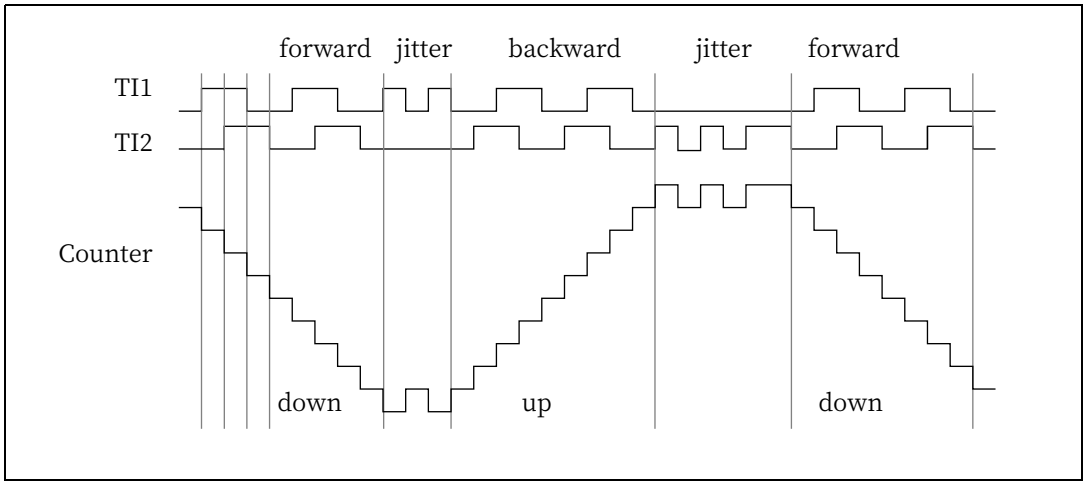


Figure 14.3-42gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=’1’).

Fig 14.3-42 Example of encoder interface mode with TI1FP1 polarity inverted



The timer, when configured in Encoder Interface mode provides information on the sensor’s current position.The user can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a real-time clock.

14.3.16 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. An example of this feature used to interface Hall sensors is given in 14.3.17

14.3.17 Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1 or TIM8) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4 or TIM5) referred to as “interfacing timer” in Figure 95. The “interfacing timer” captures the 3 timer input pins (TIMx_CH1, TIMx_CH2, and TIMx_CH3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the“interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (see Figure 78). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

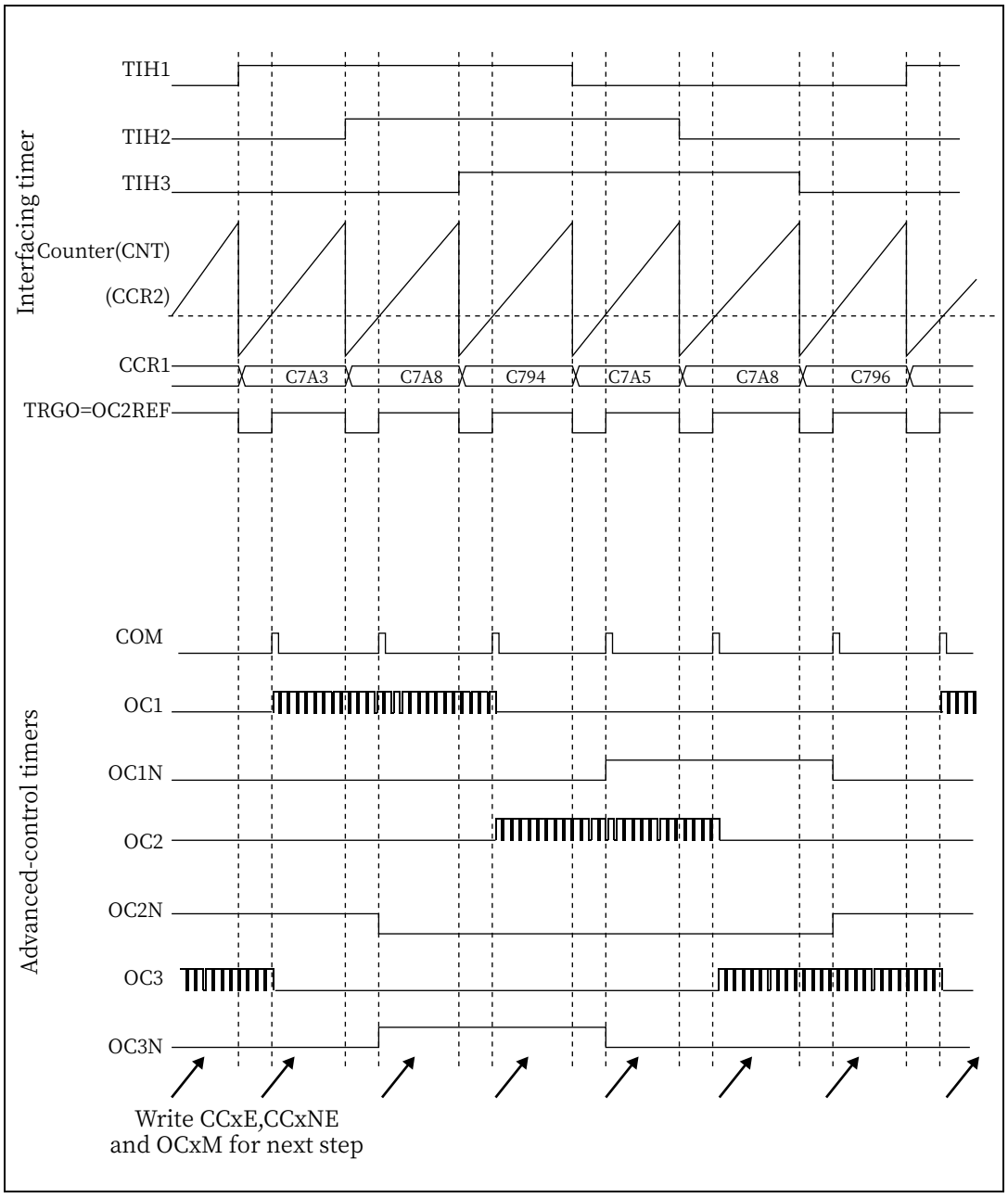
The“interfacing timer”can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1 or TIM8) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1 or TIM8) through the TRGO output.

Example: the user wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘11’. The user can also program the digital filter if needed,
- Program channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

Fig 14.3-43 Example of Hall sensor interface



14.3.18 TIMx and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

14.3.18.1 Slave mode: Reset mode

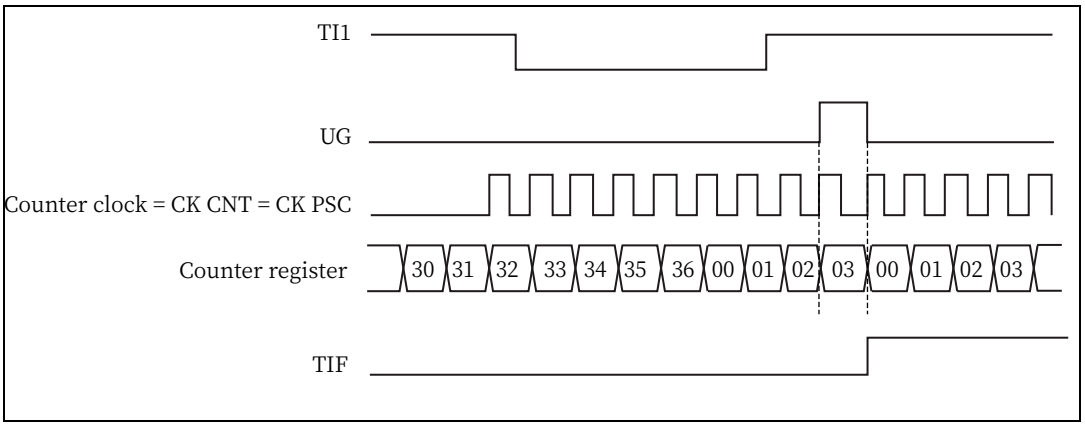
The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register). The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Fig 14.3-44 Control circuit in reset mode



14.3.18.2 Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

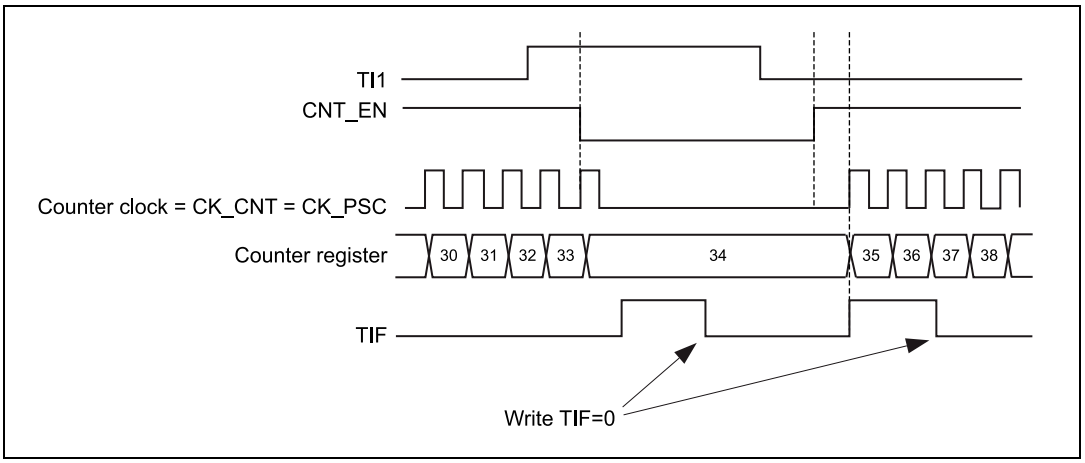
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so the user does not need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes

high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops. The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Fig 14.3-45 Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

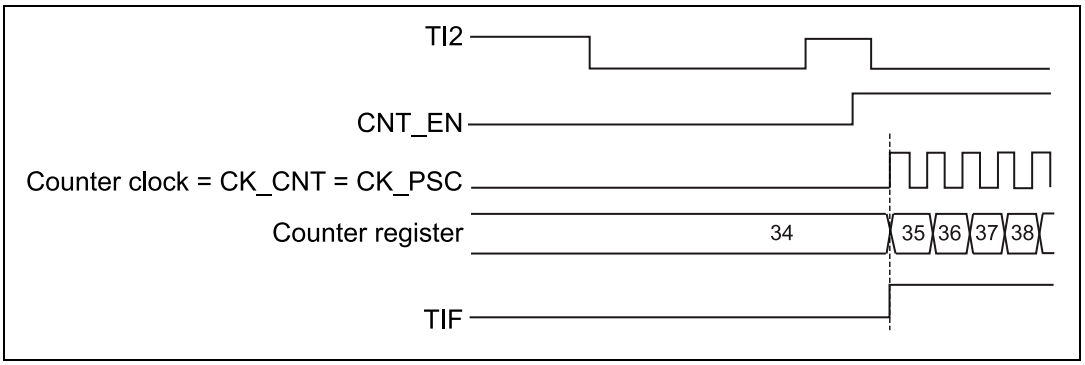
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Fig 14.3-46 Control circuit in trigger mode



14.3.18.3 Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

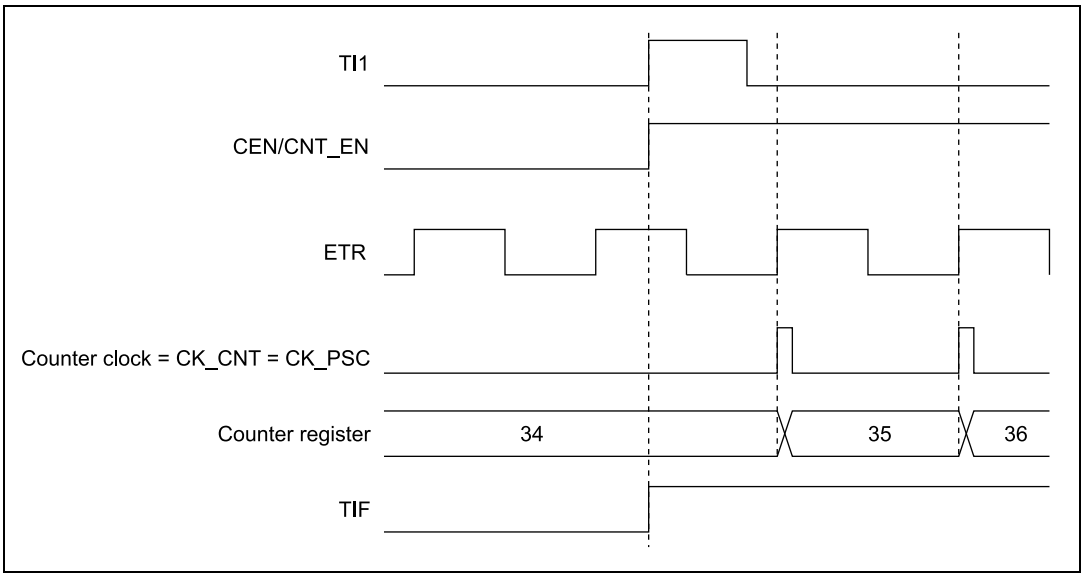
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01 in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Fig 14.3-47 Control circuit in external clock mode 2 + trigger mode



14.3.19 Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining. Refer to15.3.15

Note: *The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

14.3.20 Debug mode

When the microcontroller enters debug mode (Cortex-M3 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module.

For safety purposes, when the counter is stopped (DBG_TIMx_STOP = 1 in DBGMCU_APBx_FZ register), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

14.4 TIMx Register

Tab 14.4-1 TIMx register map

offset	Register	Reset value	Description
TIM1: TIM1_BA = 0x4001_2C00 TIM8: TIM8_BA = 0x4001_3400			
0x00	TIMx_CR1	0x0000_0000	TIM1 and TIM8 control register 1
0x04	TIMx_CR2	0x0000_0000	TIM1 and TIM8 control register 2
0x08	TIMx_SMCR	0x0000_0000	TIM1 and TIM8 slave mode control register
0x0C	TIMx_DIER	0x0000_0000	TIM1 and TIM8 DMA/interrupt enable register
0x10	TIMx_SR	0x0000_0000	TIM1 and TIM8 status register
0x14	TIMx_EGR	0x0000_0000	TIM1 and TIM8 event generation register
0x18	TIMx_CCMR1	0x0000_0000	TIM1 and TIM8 capture/compare mode register 1
0x1C	TIMx_CCMR2	0x0000_0000	TIM1 and TIM8 capture/compare mode register 2
0x20	TIMx_CCER	0x0000_0000	TIM1 and TIM8 capture/compare enable register
0x24	TIMx_CNT	0x0000_0000	TIM1 and TIM8 counter
0x28	TIMx_PSC	0x0000_0000	TIM1 and TIM8 prescaler
0x2C	TIMx_ARR	0x0000_0000	TIM1 and TIM8 auto-reload register
0x30	TIMx_RCR	0x0000_0000	TIM1 and TIM8 repetition counter register
0x34	TIMx_CCR1	0x0000_0000	TIM1 and TIM8 capture/compare register 1
0x38	TIMx_CCR2	0x0000_0000	TIM1 and TIM8 capture/compare register 2
0x3C	TIMx_CCR3	0x0000_0000	TIM1 and TIM8 capture/compare register 3
0x40	TIMx_CCR4	0x0000_0000	TIM1 and TIM8 capture/compare register 4
0x44	TIMx_BDTR	0x0000_0000	TIM1 and TIM8 break and dead-time register
0x48	TIMx_DCR	0x0000_0000	TIM1 and TIM8 DMA control register
0x4C	TIMx_DMAR	0x0000_0000	TIM1 and TIM8 DMA address for full transfer

14.4.1 TIM1 and TIM8 control register 1 (TIMx_CR1)

Register	Address offset	Access	Reset value	Description
TIMx_CR1	0x00	RW	0x0000	TIM1 and TIM8 control register 1

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						CKD[1:0]	
7	6	5	4	3	2	1	0
ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN

TIM1 and TIM8 control register 1 (TIMx_CR1)bit description

Bit	Access	Description
[15:10]	R	Reserved, must be kept at reset value.
[9:8]	RW	**CKD[1:0]:** Clock division This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx), 00: $t_DTS = t_{CK\_INT}$ 01: $t_DTS = 2*t_{CK\_INT}$ 10: $t_DTS = 4*t_{CK\_INT}$ 11: Reserved, do not program this value
[7]	RW	**ARPE:** Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
[6:5]	RW	**CMS[1:0]:** Center-aligned mode selection 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR). 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down. 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up. 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down. **Note:** *It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)*
[4]	RW	**DIR:** Direction 0: Counter used as upcounter 1: Counter used as downcounter **Note:** *This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.*

[3]	RW	**OPM:** One pulse mode enable 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit CEN)
[2]	RW	**URS:** Update request source This bit is set and cleared by software to select the UEV event sources. 0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be: • Counter overflow/underflow • Setting the UG bit • Update generation through the slave mode controller 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.
[1]	RW	**UDIS:** Update disable This bit is set and cleared by software to enable/disable UEV event generation. 0: UEV enabled. The Update (UEV) event is generated by one of the following events: • Counter overflow/underflow • Setting the UG bit • Update generation through the slave mode controller Buffered registers are then loaded with their preload values. 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.
[0]	RW	**CEN:** Counter enable 0: Counter disabled 1: Counter enabled **Note:** *External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware*

14.4.2 TIM1 and TIM8 control register 2(TIMx_CR2, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_CR2	0x04	RW	0x0000	TIM1 and TIM8 control register 2

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1
7	6	5	4	3	2	1	0
TI1S	MMS[1:0]			CCDS	CCUS	Reserved	CCPC

TIM1 and TIM8 control register 2(TIMx_CR2)bit description

Bit	Access	Description
[15]	R	Reserved, must be kept at reset value.
[14]	RW	**OIS4:** Output Idle state 4 (OC4 output) refer to OIS1 bit
[13]	RW	**OIS3N:** Output Idle state 3 (OC3N output) refer to OIS1N bit
[12]	RW	**OIS3:** Output Idle state 3 (OC3 output) refer to OIS1 bit
[11]	RW	**OIS2N:** Output Idle state 2 (OC2N output) refer to OIS1N bit
[10]	RW	**OIS2:** Output Idle state 2 (OC2 output) refer to OIS1 bit
[9]	RW	**OIS1N:** Output Idle state 1 (OC1N output) 0: OC1N=0 after a dead-time when MOE=0 1: OC1N=1 after a dead-time when MOE=0 **Note:** *This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*
[8]	RW	**OIS1:** Output Idle state 1 (OC1 output) 0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0 1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0 **Note:** *This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register)*
[7]	RW	**TI1S:** TI1 selection 0: The TIMx_CH1 pin is connected to TI1 input 1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

<table>
<tr><td>[6:4]</td><td>RW</td><td>MMS[2:0]: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
<table>
<tr><th>MMS</th><th>mode</th><th>Description</th></tr>
<tr><td>000:</td><td>Reset</td><td>the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.</td></tr>
<tr><td>001:</td><td>Enable</td><td>the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).</td></tr>
<tr><td>010:</td><td>Update</td><td>The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.</td></tr>
<tr><td>011:</td><td>Compare Pulse</td><td>The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)</td></tr>
<tr><td>100:</td><td>Compare</td><td>OC1REF signal is used as trigger output (TRGO)</td></tr>
<tr><td>101:</td><td>Compare</td><td>OC2REF signal is used as trigger output (TRGO)</td></tr>
<tr><td>110:</td><td>Compare</td><td>OC3REF signal is used as trigger output (TRGO)</td></tr>
<tr><td>111:</td><td>Compare</td><td>OC4REF signal is used as trigger output (TRGO)</td></tr>
</table></td></tr>
<tr><td>[3]</td><td>RW</td><td>CCDS: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs</td></tr>
<tr><td>[2]</td><td>RW</td><td>Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI。
Note: This bit acts only on channels that have a complementary output.</td></tr>
<tr><td>[1]</td><td>R</td><td>Reserved, must be kept at reset value.</td></tr>
<tr><td>[0]</td><td>RW</td><td>CCDS: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.</td></tr>
</table>

14.4.3 TIM1 and TIM8 slave mode control register(TIMx_SMCR, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_SMCR	0x08	RW	0x0000_0000	TIM1 and TIM8 slave mode control register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ETP	ECE	ETPS[1:0]		ETF[3:0]			
7	6	5	4	3	2	1	0
MSM	TS[2:0]			Reserved	SMS[2:0]		

TIM1 and TIM8 slave mode control register(TIMx_SMCR)bit description

Bit	Access	Description
[15]	RW	**ETP:** External trigger polarity This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations 0: ETR is non-inverted, active at high level or rising edge. 1: ETR is inverted, active at low level or falling edge.
[14]	RW	**ECE:** External clock enable This bit enables External clock mode 2. 0: External clock mode 2 disabled 1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal. **Note:** 1. *Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).* 2. *It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).* 3. *If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*
[13:12]	RW	**ETPS[1:0]:** External trigger prescaler External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks. 00: Prescaler OFF 01: ETRP frequency divided by 2 10: ETRP frequency divided by 4 11: ETRP frequency divided by 8

[11:8]	RW	**ETF[3:0]:** External trigger filter This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:。 0000: No filter, sampling is done at f_{DTS} 0001: $f_{SAMPLING} = f_{CK\_INT}$, N=2 0010: $f_{SAMPLING} = f_{CK\_INT}$, N = 4 0011: $f_{SAMPLING} = f_{CK\_INT}$, N = 8 0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6 0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8 0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6 0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8 1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6 1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8 1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5 1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6 1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8 1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5 1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6 1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8
[7]	RW	**MSM:** Master/slave mode 0: No action 1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.
[6:4]	RW	**TS[2:0]:** Trigger selection This bit-field selects the trigger input to be used to synchronize the counter. 000: Internal Trigger 0(ITR0) 001: Internal Trigger 1(ITR1) 010: Internal Trigger 2(ITR2) 011: Internal Trigger 3(ITR3) 100: TI1 Edge Detector (TI1F_ED) 101: Filtered Timer Input 1(TI1FP1) 110: Filtered Timer Input 2(TI2FP2) 111: External Trigger input(ETRF) See Table 14.4-5 for more details on ITRx meaning for each Timer **Note:** *These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*
[3]	R	Reserved, must be kept at reset value.

<table>
<tr><td>[2:0]</td><td>RW</td><td>SMS[2:0]: Slave mode selection
When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.)
<table>
<tr><th>SMS</th><th>Mode</th><th>Description</th></tr>
<tr><td>000:</td><td>Slave mode disabled</td><td>if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.</td></tr>
<tr><td>001:</td><td>Encoder mode 1</td><td>Counter counts up/down on TI2FP1 edge depending on TI1FP2 level.</td></tr>
<tr><td>010:</td><td>Encoder mode 2</td><td>Counter counts up/down on TI1FP2 edge depending on TI2FP1 level.</td></tr>
<tr><td>011:</td><td>Encoder mode 3</td><td>Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input</td></tr>
<tr><td>100:</td><td>Reset Mode</td><td>Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.</td></tr>
<tr><td>101:</td><td>Gated Mode</td><td>The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.</td></tr>
<tr><td>110:</td><td>Trigger Mode</td><td>The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.</td></tr>
<tr><td>111:</td><td>External Clock Mode 1</td><td>Rising edges of the selected trigger (TRGI) clock the counter.</td></tr>
</table>
Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=’100’). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.
The clock of the slave timer must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.</td></tr>
</table>

Tab 14.4-5 TIMx Internal trigger connection

Slave TIM	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

14.4.4 TIM1 and TIM8 DMA/interrupt enable register(TIMx_DIER, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_DIER	0x0C	RW	0x0000_0000	TIM1 and TIM8 DMA/interrupt enable register (x = 1,8)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE
7	6	5	4	3	2	1	0
BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE

TIM1 and TIM8 DMA/interrupt enable register(TIMx_DIER)bit description

Bit	Access	Description
[15]	R	Reserved, must be kept at reset value
[14]	RW	**TDE:** Trigger DMA request enable 0: Trigger DMA request disabled 1: Trigger DMA request enabled
[13]	RW	**COMDE:** COM DMA request enable 0: COM DMA request disabled 1: COM DMA request enabled
[12]	RW	**CC4DE:** Capture/Compare 4 DMA request enable 0: CC4 DMA request disabled 1: CC4 DMA request enabled
[11]	RW	**CC3DE:** Capture/Compare 3 DMA request enable 0: CC3 DMA request disabled 1: CC3 DMA request enabled
[10]	RW	**CC2DE:** Capture/Compare 2 DMA request enable 0: CC2 DMA request disabled 1: CC2 DMA request enabled
[9]	RW	**CC1DE:** Capture/Compare 1 DMA request enable 0: CC1 DMA request disabled 1: CC1 DMA request enabled
[8]	RW	**UDE:** Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled
[7]	RW	**BIE:** Break interrupt enable 0: Break interrupt disabled 1: Break interrupt enabled
[6]	RW	**TIE:** Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
[5]	RW	**COMIE:** COM interrupt enable 0: COM interrupt disabled 1: COM interrupt enabled
[4]	RW	**CC4IE:** Capture/Compare 4 interrupt enable 0: CC4 interrupt disabled 1: CC4 interrupt enabled
[3]	RW	**CC3IE:** Capture/Compare 3 interrupt enable 0: CC3 interrupt disabled 1: CC3 interrupt enabled

[2]	RW	**CC2IE:** Capture/Compare 2 interrupt enable 0: CC2 interrupt disabled 1: CC2 interrupt enabled
[1]	RW	**CC1IE:** Capture/Compare 1 interrupt enable 0: CC1 interrupt disabled 1: CC1 interrupt enabled
[0]	RW	**UIE:** Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

14.4.5 TIM1 and TIM8 status register(TIMx_SR, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_SR	0x10	RC_W0	0x0000_0000	TIM1 and TIM8 status register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved
7	6	5	4	3	2	1	0
BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF

TIM1 and TIM8 status register(TIMx_SR)bit description

Bit	Access	Description
[15:13]	R	Reserved, must be kept at reset value.
[12]	RC_W0	**CC4OF:** Capture/Compare 4 overcapture flag refer to CC1OF description
[11]	RC_W0	**CC3OF:** Capture/Compare 3 overcapture flag refer to CC1OF description
[10]	RC_W0	**CC2OF:** Capture/Compare 2 overcapture flag refer to CC1OF description
[9]	RC_W0	**CC1OF:** Capture/Compare 1 overcapture flag This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’. 0: No overcapture has been detected. 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set
[8]	R	Reserved, must be kept at reset value
[7]	RC_W0	**BIF:** Break interrupt flag This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active. 0: No break event occurred. 1: An active level has been detected on the break input.

[6]	RC_W0	**TIF:** Trigger interrupt flag This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software. 0: No trigger event occurred. 1: Trigger interrupt pending.
[5]	RC_W0	**COMIF:** COM interrupt flag This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software. 0: No COM event occurred. 1: COM interrupt pending.
[4]	RC_W0	**CC4IF:** Capture/Compare 4 interrupt flag refer to CC1IF description
[3]	RC_W0	**CC3IF:** Capture/Compare 3 interrupt flag refer to CC1IF description
[2]	RC_W0	**CC2IF:** Capture/Compare 2 interrupt flag refer to CC1IF description
[1]	RC_W0	**CC1IF:** Capture/Compare 1 interrupt flag **If channel CC1 is configured as output:** This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software. 0: No match. 1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode) **If channel CC1 is configured as input:** This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register. 0: No input capture occurred 1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)
[0]	RC_W0	**UIF:** Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update occurred. 1: Update interrupt pending. This bit is set by hardware when the registers are updated: • At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register. • When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register. • When CNT is reinitialized by a trigger event (TIMx_SMCR), if URS=0 and UDIS=0 in the TIMx_CR1 register.

14.4.6 TIM1 and TIM8 event generation register(TIMx_EGR, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_EGR	0x14	W	0x0000_0000	TIM1 and TIM8 event generation register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG

TIM1 and TIM8 event generation register(TIMx_EGR)bit description

Bit	Access	Description
[31:8]	R	Reserved, must be kept at reset value
[7]	W	**BG:** Break generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.
[6]	W	**TG:** Trigger generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.
[5]	W	**COMIF:** Capture/Compare control update generation This bit can be set by software, it is automatically cleared by hardware 0: No action 1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits **Note:** *This bit acts only on channels having a complementary output.*
[4]	W	**CC4G:** Capture/Compare 4 generation refer to CC1G description
[3]	W	**CC3G:** CC3G: Capture/Compare 3 generation refer to CC1G description
[2]	W	**CC2G:** CC2G: Capture/Compare 2 generation refer to CC1G description

[1]	W	**CC1G:** Capture/Compare 1 generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: A capture/compare event is generated on channel 1: **If channel CC1 is configured as output:** CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled. **If channel CC1 is configured as input:** The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high
[0]	W	**UG:** Update generation This bit can be set by software, it is automatically cleared by hardware. 0: No action 1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

14.4.7 TIM1 and TIM8 capture/compare mode register 1(TIMx_CCMR1, x = 1,8)

Note: *The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So the user must take care that the same bit can have a different meaning for the input stage and for the output stage.*

Register	Address offset	Access	Reset value	Description
TIMx_CCMR1	0x18	RW	0x0000_0000	TIM1 and TIM8 capture/compare mode register 1

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]	
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]	
7	6	5	4	3	2	1	0
IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	

Input capture mode

TIM1 and TIM8 capture/compare mode register 1(TIMx_CCMR1)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.
[15:12]	RW	**IC2F[3:0]:** Input capture 2 filter
[11:10]	RW	**IC2PSC[1:0]:** Input capture 2 prescaler

[9:8]	RW	**CC2S[1:0]:** Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).*
[7:4]	RW	**IC1F[3:0]:** Input capture 1 filter This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output: 0000: No filter, sampling is done at f_{DTS} 0001: $f_{SAMPLING} = f_{CK\_INT}$, N=2 0010: $f_{SAMPLING} = f_{CK\_INT}$, N = 4 0011: $f_{SAMPLING} = f_{CK\_INT}$, N = 8 0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6 0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8 0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6 0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8 1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6 1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8 1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5 1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6 1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8 1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5 1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6 1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8
[3:2]	RW	**IC1PSC[1:0]:** Input capture 1 prescaler This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register). 00: no prescaler, capture is done each time an edge is detected on the capture input 01: capture is done once every 2 events 10: capture is done once every 4 events 11: capture is done once every 8 events
[1:0]	RW	**CC1S[1:0]:** Capture/Compare 1 Selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER)*

Output compare mode:

TIM1 and TIM8 capture/compare mode register 1(TIMx_CCMR1)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.
[15]	RW	**OC2CE:** Output Compare 2 clear enable
[14:12]	RW	**OC2M[2:0]:** Output Compare 2 mode
[11]	RW	**OC2PE:** Output Compare 2 preload enable
[10]	RW	**OC2FE:** Output Compare 2 fast enable

[9:8]	RW	**CC2S[1:0]:** Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register) **Note:** *CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).*
[7]	RW	**OC1CE:** Output Compare 1 clear enable OC1CE: Output Compare 1 Clear Enable 0: OC1Ref is not affected by the ETRF Input 1: OC1Ref is cleared as soon as a High level is detected on ETRF input
[6:4]	RW	**OC1M[2:0]:** Output Compare 1 mode These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits. 000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base). 001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1). 010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1). 011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1. 100: Force inactive level - OC1REF is forced low. 101: Force active level - OC1REF is forced high. 110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF= '0') as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF='1'). 111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive. **Note:** *1.These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).* *2. In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*
[3]	RW	**OC1PE:** Output Compare 1 preload enable 0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately. 1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event. **Note:** *1.These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).* *2.The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

[2]	RW	**OC1FE:** Output Compare 1 fast enable This bit is used to accelerate the effect of an event on the trigger in input on the CC output. 0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles. 1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.
[1:0]	RW	**CC1S[1:0]:** Capture/Compare 1 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER)*

14.4.8 TIM1 and TIM8 capture/compare mode register 2(TIMx_CCMR2, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_CCMR2	0x1C	RW	0x0000_0000	TIM1 and TIM8 capture/compare mode register 2

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]	
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]	
7	6	5	4	3	2	1	0
IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	

Input capture mode

TIM1 and TIM8 capture/compare mode register 2(TIMx_CCMR2)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.
[15:12]	RW	**IC4F[3:0]:** Input capture 4 filter
[11:10]	RW	**IC4PSC[1:0]:** Input capture 4 prescaler

[9:8]	RW	**CC4S[1:0]:** Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).*
[7:4]	RW	**IC3F[3:0]:** Input capture 3 filter
[3:2]	RW	**IC3PSC[1:0]:** Input capture 3 prescaler
[1:0]	RW	**CC3S[1:0]:** Capture/compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).*

Output compare mode:

TIM1 and TIM8 capture/compare mode register 2(TIMx_CCMR2)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.
[15]	RW	**OC4CE:** Output compare 4 clear enable
[14:12]	RW	**OC4M[2:0]:** Output compare 4 mode
[11]	RW	**OC4PE:** Output compare 4 preload enable
[10]	RW	**OC4FE:** Output compare 4 fast enable
[9:8]	RW	**CC4S[1:0]:** Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).*
[7]	RW	**OC3CE:** Output compare 3 clear enable
[6:4]	RW	**OC3M[2:0]:** Output compare 3 mode
[3]	RW	**OC3PE:** Output compare 3 preload enable
[2]	RW	**OC3FE:** Output compare 3 fast enable
[1:0]	RW	**CC3S[1:0]:** Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).*

14.4.9 TIM1 and TIM8 capture/compare enable register(TIMx_CCER, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_CCER	0x20	RW	0x0000_0000	TIM1 and TIM8 capture/compare enable register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E
7	6	5	4	3	2	1	0
CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E

TIM1 and TIM8 capture/compare enable register(TIMx_CCER)bit description

Bit	Access	Description
[31:14]	-	Reserved, must be kept at reset value.
[13]	RW	**CC4P:** Capture/Compare 4 output polarity refer to CC1P description
[12]	RW	**CC4E:** Capture/Compare 4 output enable refer to CC1E description
[11]	RW	**CC3NP:** Capture/Compare 3 complementary output polarity refer to CC1NP description
[10]	RW	**CC3NE:** Capture/Compare 3 complementary output enable refer to CC1NE description
[9]	RW	**CC3P:** Capture/Compare 3 output polarity refer to CC1P description
[8]	RW	**CC3E:** Capture/Compare 3 output enable refer to CC1E description
[7]	RW	**CC2NP:** Capture/Compare 2 complementary output polarity refer to CC1NP description
[6]	RW	**CC2NE:** Capture/Compare 2 complementary output enable refer to CC1NE description
[5]	RW	**CC2P:** Capture/Compare 2 output polarity refer to CC1P description
[4]	RW	**CC2E:** Capture/Compare 2 output enable refer to CC1E description
[3]	RW	**CC1NP:** Capture/Compare 1 complementary output polarity 0: OC1N active high. 1: OC1N active low. **Note:** *This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=”00”(the channel is configured in output).*
[2]	RW	**CC1NE:** Capture/Compare 1 complementary output enable 0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits. 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

[1]	RW	**CC1P:** Capture/Compare 1 output polarity **CC1 channel configured as output:** 0: OC1 active high 1: OC1 active low **CC1 channel configured as input:** This bit selects whether IC1 or IC1 is used for trigger or capture operations. 0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted. 1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted. **Note:** *This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*
[0]	RW	**CC1E:** Capture/Compare 1 output enable **CC1 channel configured as output:** 0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits. 1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits. **CC1 channel configured as input:** This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not. 0: Capture disabled. 1: Capture enabled.

Tab 14.4-14 Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states	
MOE	**OSSI**	**OSSR**	**CCxE**	**CCxNE**	**OCx output state**	**OCxN output state**
1	x	0	0	0	Output Disabled (not driven by the timer)OCx=0, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	0	1	Output Disabled (not driven by the timer)OCx=0, OCx_EN=0	OCxREF + Polarity OCxN = (OCxREF xor CCxNP), OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx = (OCxREF xor CCxP), OCx_EN=1	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	1	1	OCxREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF + Polarity + dead-time OCxN_EN=1
		1	0	0	Output Disabled (not driven by the timer)OCx=CCxP, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP, OCx_EN=1	OCxREF + Polarity OCxN = (OCxREF xor CCxNP), OCxN_EN=1
		1	1	0	OCxREF + Polarity OCx = (OCxREF xor CCxP), OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCxREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF + Polarity + dead-time OCxN_EN=1
0	0	x	0	0	Output Disabled (not driven by the timer) Asynchronously: OCx=CCxP , OCx_EN=0 , OCxN=CCxNP , OCxN_EN=0 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0		
	1		0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX	
	1		1	0		
	1		1	1		

When both outputs of a channel are not used (CCxE = CCxNE = 0), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: *The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIOand AFIO registers.*

14.4.10 TIM1 and TIM8 counter(TIMx_CNT, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_CNT	0x24	RW	0x0000_0000	TIM1 and TIM8 counter

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

TIM1 and TIM8 counter(TIMx_CNT)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.
[15:0]	RW	**CNT[15:0]:** Counter value

14.4.11 TIM1 and TIM8 prescaler(TIMx_PSC, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_PSC	0x28	RW	0x0000_0000	TIM1 and TIM8 prescaler

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

TIM1 and TIM8 prescaler(TIMx_PSC)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.
[15:0]	RW	**PSC[15:0]:** Prescaler value The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1). PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

14.4.12 TIM1 and TIM8 auto-reload register(TIMx_ARR, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_ARR	0x2C	RW	0x0000_0000	TIM1 and TIM8 auto-reload register

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

TIM1 and TIM8 auto-reload register(TIMx_ARR)bit description

Bit	Access	Description
[31:0]	RW	**ARR[31:0]:** Auto-reload value ARR is the value to be loaded in the actual auto-reload register. Refer to Section 14.3.1: Time-base unit for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.

14.4.13 TIM1 and TIM8 repetition counter register(TIMx_RCR, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_RCR	0x30	RW	0x0000_0000	TIM1 and TIM8 repetition counter register (x = 1,8)

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

TIM1 and TIM8 repetition counter register(TIMx_RCR)bit description

Bit	Access	Description
[31:8]	R	Reserved, must be kept at reset value.
[7:0]	RW	**RFP[7:0]:** Repetition counter value These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable. Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event. It means in PWM mode (REP+1) corresponds to: • the number of PWM periods in edge-aligned mode • the number of half PWM period in center-aligned mode

14.4.14 TIM1 and TIM8 capture/compare register 1 (TIMx_CCR1, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_CCR1	0x34	RW	0x0000_0000	TIM1 and TIM8 capture/compare register 1 (x = 1,8)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CCR1[15:8]							
7	6	5	4	3	2	1	0
CCR1[7:0]							

TIM1 and TIM8 capture/compare register 1 (TIMx_CCR1)bit description

Bit	Access	Description
[31:16]	-	TIM1 and TIM8 capture/compare register 1
[15:0]	RW	**CCR1[15:0]:** Capture/Compare 1 value **If channel CC1 is configured as output:** CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output. **If channel CC1 is configured as input:** CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

14.4.15 TIM1 and TIM8 capture/compare register 2(TIMx_CCR2, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_CCR2	0x38	RW	0x0000_0000	TIM1 and TIM8 capture/compare register 2

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CCR2[15:8]							
7	6	5	4	3	2	1	0
CCR2[7:0]							

TIM1 and TIM8 capture/compare register 2(TIMx_CCR2)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.

[15:0]	RW	**CCR2[15:0]:** Capture/Compare 2 value **If channel CC2 is configured as output:** CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output. **If channel CC2 is configured as input:** CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

14.4.16 TIM1 and TIM8 capture/compare register 3 (TIMx_CCR3, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_CCR3	0x3C	RW	0x0000_0000	TIM1 and TIM8 capture/compare register 3

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CCR3[15:8]							
7	6	5	4	3	2	1	0
CCR3[7:0]							

TIM1 and TIM8 capture/compare register 3 (TIMx_CCR3)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**CCR3[15:0]:** Capture/Compare value **If channel CC3 is configured as output:** CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output. **If channel CC3 is configured as input:** CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

14.4.17 TIM1 and TIM8 capture/compare register 4(TIMx_CCR4, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_CCR4	0x40	RW	0x0000_0000	TIM1 and TIM8 capture/compare register 4

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CCR4[15:8]							
7	6	5	4	3	2	1	0
CCR4[7:0]							

TIM1 and TIM8 capture/compare register 4(TIMx_CCR4)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**CCR4[15:0]:** Capture/Compare value **If channel CC4 is configured as output:** CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output. **If channel CC4 is configured as input:** CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR3 register is read-only and cannot be programmed.

14.4.18 TIM1 and TIM8 break and dead-time register(TIMx_BDTR, x = 1,8)

Note: *As the bits AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.*

Register	Address offset	Access	Reset value	Description
TIMx_BDTR	0x44	RW	0x0000_0000	TIM1 and TIM8 break and dead-time register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]	
7	6	5	4	3	2	1	0
DTG[7:0]							

TIM1 and TIM8 break and dead-time register(TIMx_BDTR)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15]	RW	**MOE:** Main output enable This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output. 0: OC and OCN outputs are disabled or forced to idle state. 1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register). See OC/OCN enable description for more details((Section 14.4.9:TIMx_CCER).)
[14]	RW	**AOE:** Automatic output enable 0: MOE can be set only by software 1: MOE can be set by software or automatically at the next update event (if the break input is not be active) **Note:** *This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*
[13]	RW	**BKP:** Break polarity 0: Break input BRK is active low 1: Break input BRK is active high **Note:** *1.This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).* *2. Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*
[12]	RW	**BKE:** Break enable 0: Break inputs (BRK and CSS clock failure event) disabled 1; Break inputs (BRK and CSS clock failure event) enabled **Note:** *1. This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).* *2. Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

[11]	RW	**OSSR:** Off-state selection for Run mode This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer. See OC/OCN enable description for more details (Section 14.4.9:TIMx_CCER). 0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0). 1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1. Then, OC/OCN enable output signal=1 **Note:** *This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register)*
[10]	RW	**OSSI:** Off-state selection for Idle mode This bit is used when MOE=0 on channels configured as outputs. See OC/OCN enable description for more details (Section 14.4.9:TIMx_CCER). 0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0). 1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1 **Note:** *This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).*
[9:8]	RW	**LOCK[1:0]:** Lock configuration These bits offer a write protection against software errors. 00: LOCK OFF - No bit is write protected. 01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written. 10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written. 11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written. **Note:** *The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.*
[7:0]	RW	**DTG[7:0]:** Dead-time generator setup This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration. DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with t_{dtg} =T_{DTS} . DTG[7:5]=10x => DT=(64+DTG[5:0])x t_{dtg} with T_{dtg} =2x t_{DTS} . DTG[7:5]=110 => DT=(32+DTG[4:0])x t_{dtg} with T_{dtg} =8x t_{DTS} . DTG[7:5]=111 => DT=(32+DTG[4:0])x t_{dtg} with T_{dtg} =16x t_{DTS} . Example if T_{DTS} = 125ns (8MHz), dead-time possible values are: 0 to 15875 ns by 125 ns steps, 16 us to 31750 ns by 250 ns steps, 32 us to 63 us by 1 us steps, 64 us to 126 us by 2 us steps **Note:** *This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

14.4.19 TIM1 and TIM8 DMA control register(TIMx_DCR, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_DCR	0x48	RW	0x0000_0000	TIM1 and TIM8 DMA control register (x = 1,8)

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

TIM1 and TIM8 DMA control register(TIMx_DCR)bit description

Bit	Access	Description
[31:13]	R	Reserved, must be kept at reset value
[12:8]	RW	**DBL[4:0]:** DMA burst length This 5-bit vector defines the number of DMA transfers (the timer detects a burst transfer when a read or a write access to the TIMx_DMAR register address is performed). the TIMx_DMAR address: 00000 ~ 10001: 0 ~ 18 transfers
[7:5]	R	Reserved, must be kept at reset value.
[4:0]	RW	**DBA[4:0]:** DMA base address This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register: 00000: TIMx_CR1 00001: TIMx_CR2 00010: TIMx_SMCR ... ... Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

14.4.20 TIM1 and TIM8 DMA address for full transfer(TIMx_DMAR, x = 1,8)

Register	Address offset	Access	Reset value	Description
TIMx_DMAR	0x4C	RW	0x0000_0000	TIM1 and TIM8 DMA address for full transfer

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

TIM1 and TIM8 DMA address for full transfer(TIMx_DMAR)bit description

Bit	Access	Description
[31:0]	RW	**DMAB[15:0]:** DMA register for burst accesses A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4 where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR). **Example of how to use the DMA burst feature** In this example the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) with the DMA transferring half words into the CCRx registers. This is done in the following steps: 1. Configure the corresponding DMA channel as follows: • DMA channel peripheral address is the DMAR register address • DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers. • Number of data to transfer = 3 (See note below). • Circular mode disabled. 2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE. 3. Enable the TIMx update DMA request (set the UDE bit in the DIER register). 4. Enable TIMx 5. Enable the DMA channel **Note:** *This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.*

15 General-purpose timers (TIM2 to TIM5)

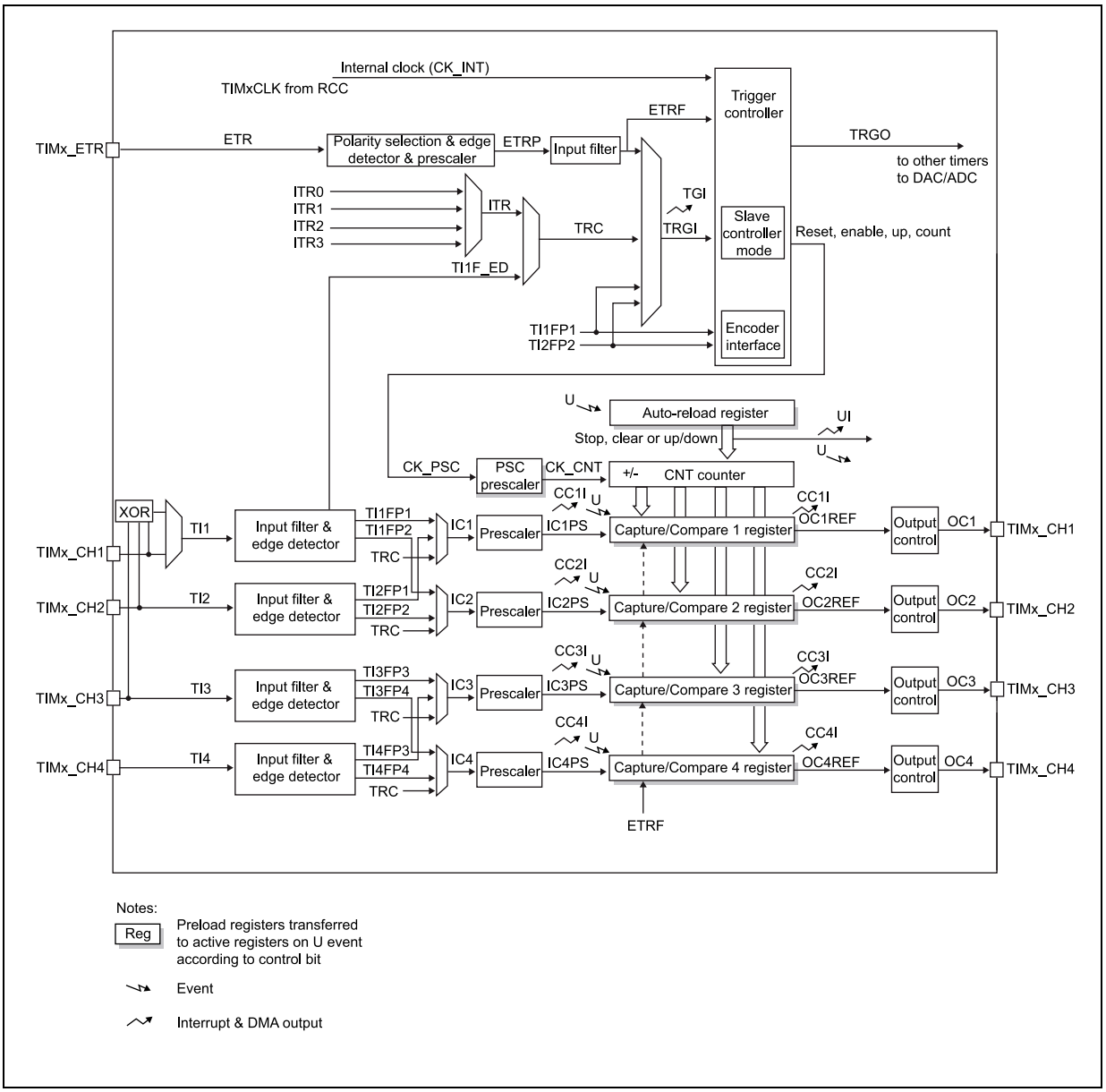
15.1 TIM2 to TIM5 introduction

The general-purpose timers consist of a 32-bit auto-reload counter driven by a programmable prescaler.
They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).
Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.
The timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 15.3.15

15.2 TIMx main features

- 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Fig 15.2-1 General-purpose timer block diagram



15.3 TIMx functional description

15.3.1 Time-base unit

The main block of the programmable timer is a 32-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC):

- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Fig 15.3-1 Counter timing diagram with prescaler division change from 1 to 2

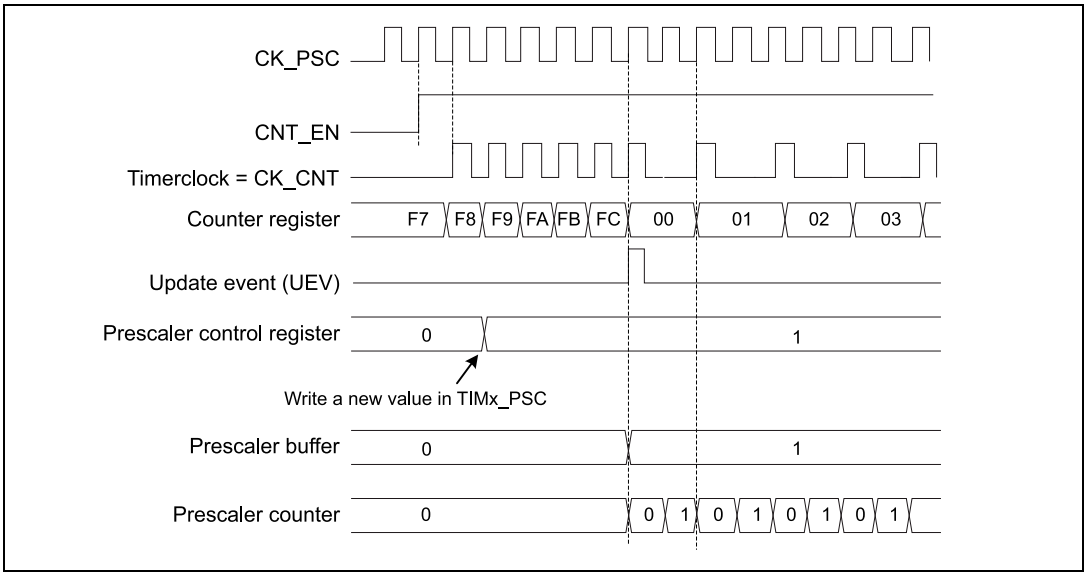
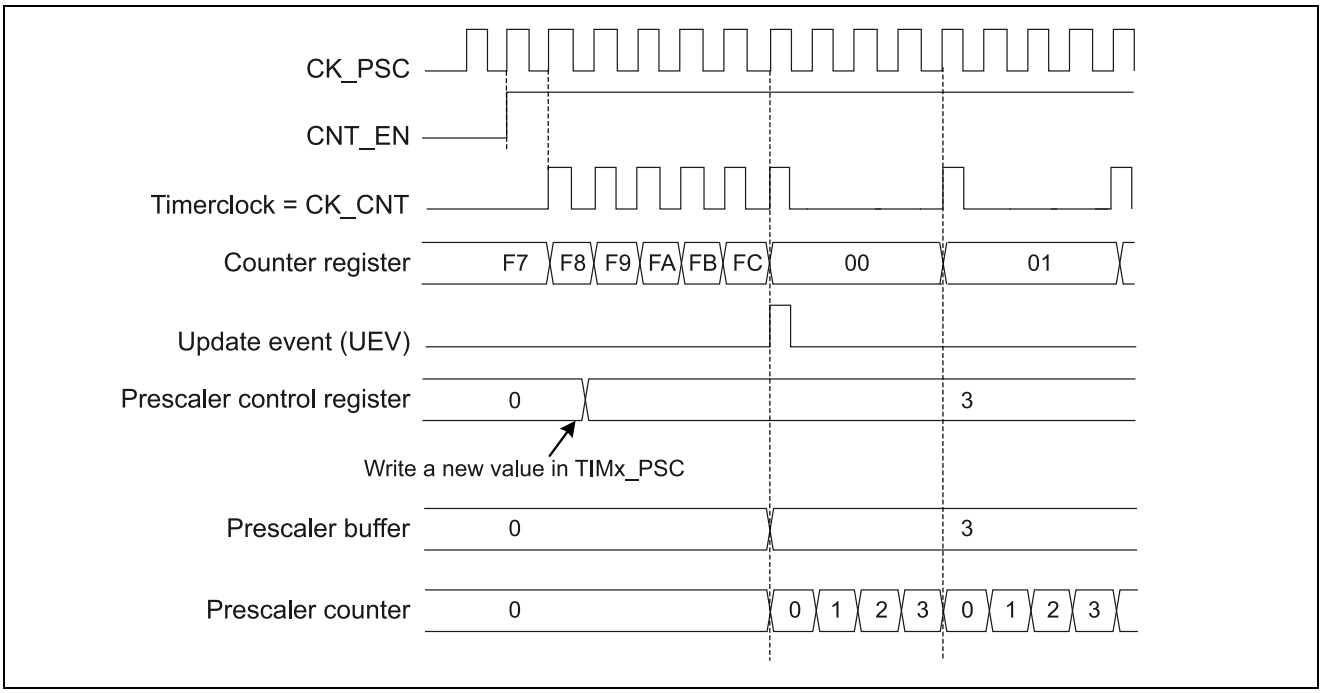


Fig 15.3-2 Counter timing diagram with prescaler division change from 1 to 4



15.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

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

Fig 15.3-3 Counter timing diagram, internal clock divided by 1

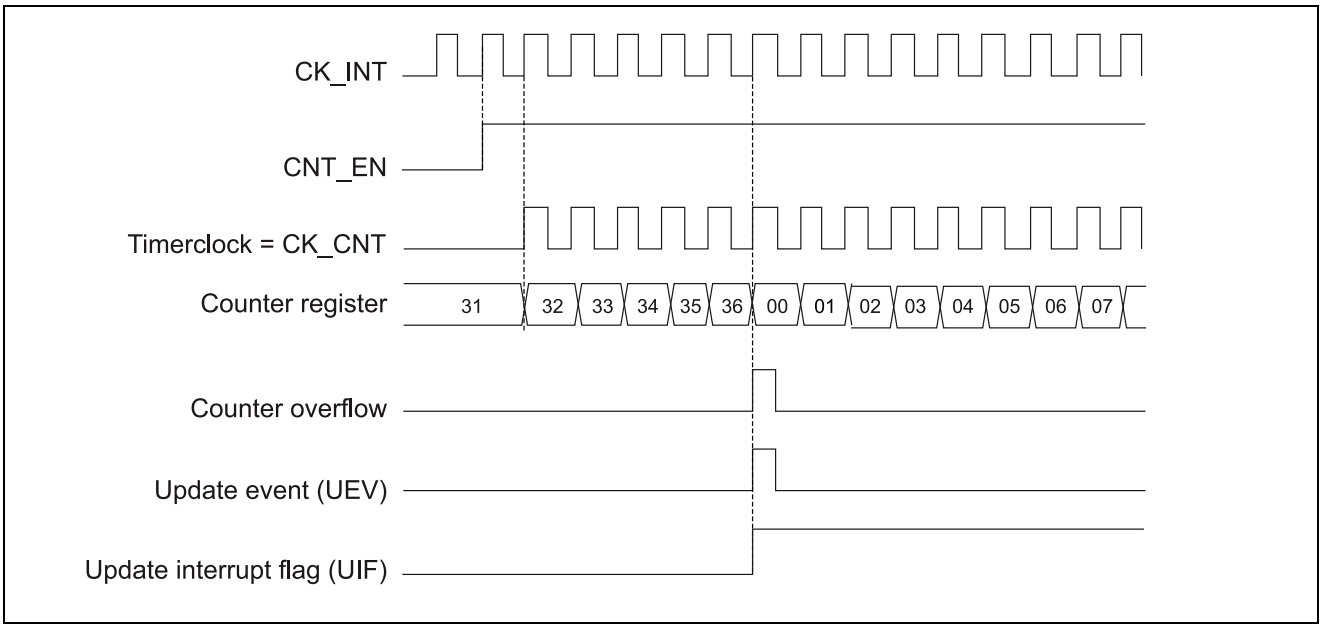


Fig 15.3-4 Counter timing diagram, internal clock divided by 2

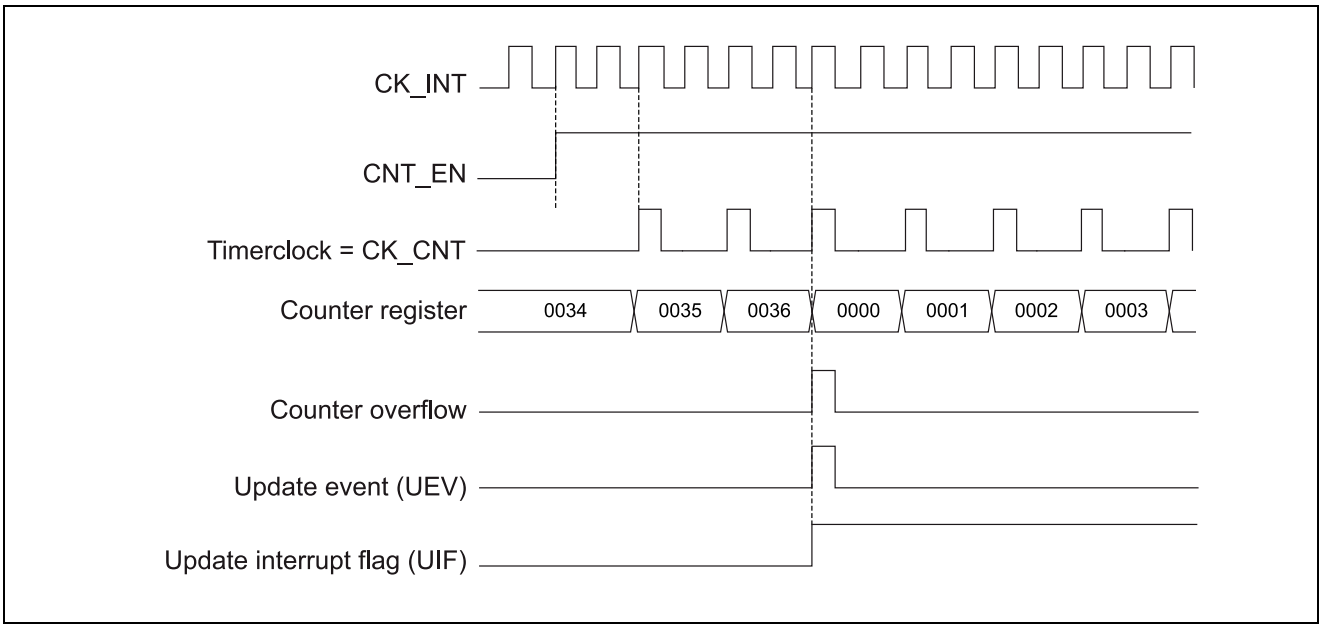


Fig 15.3-5 Counter timing diagram, internal clock divided by 4

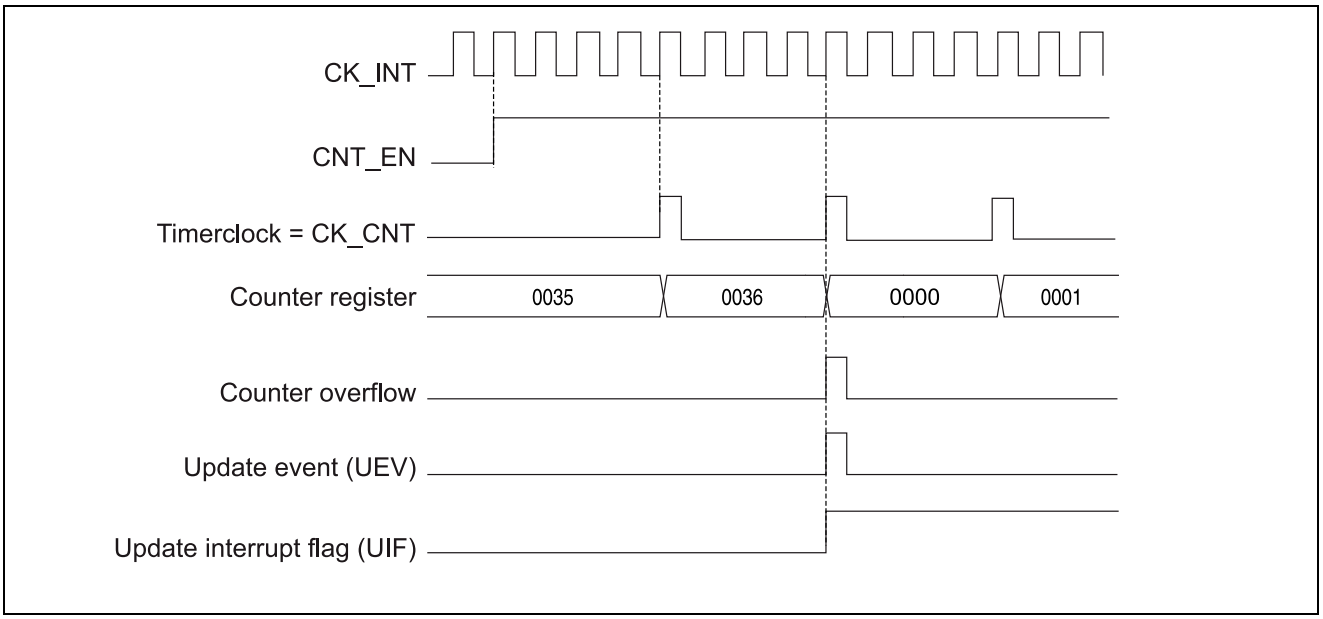


Fig 15.3-6 Counter timing diagram, internal clock divided by N

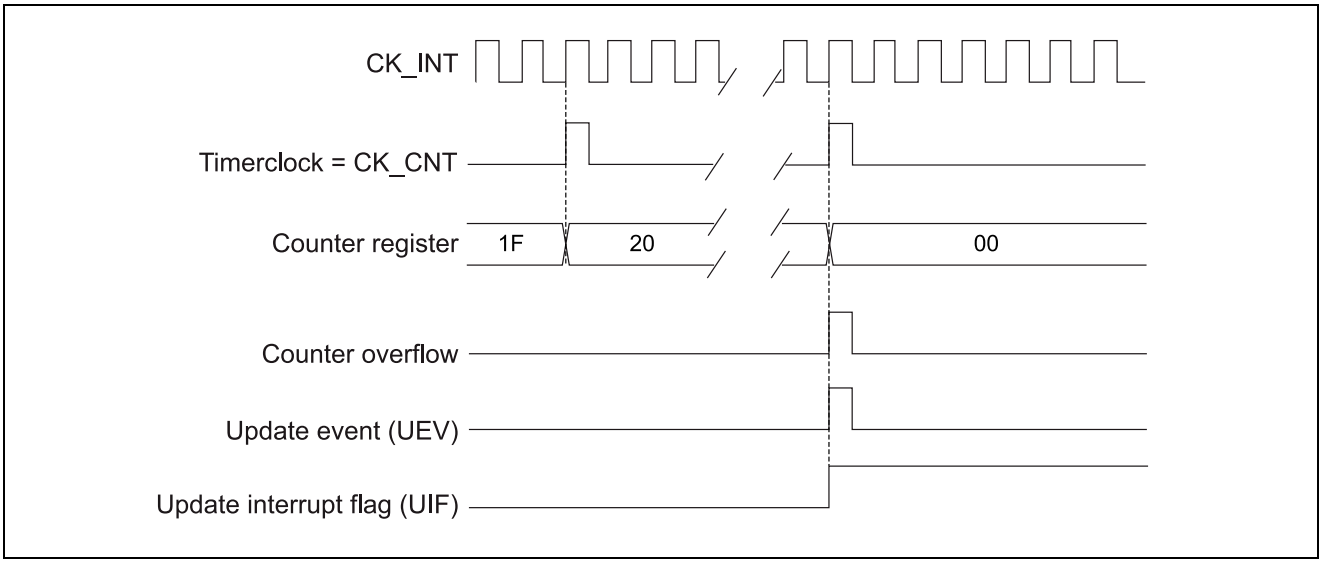


Fig 15.3-7 Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

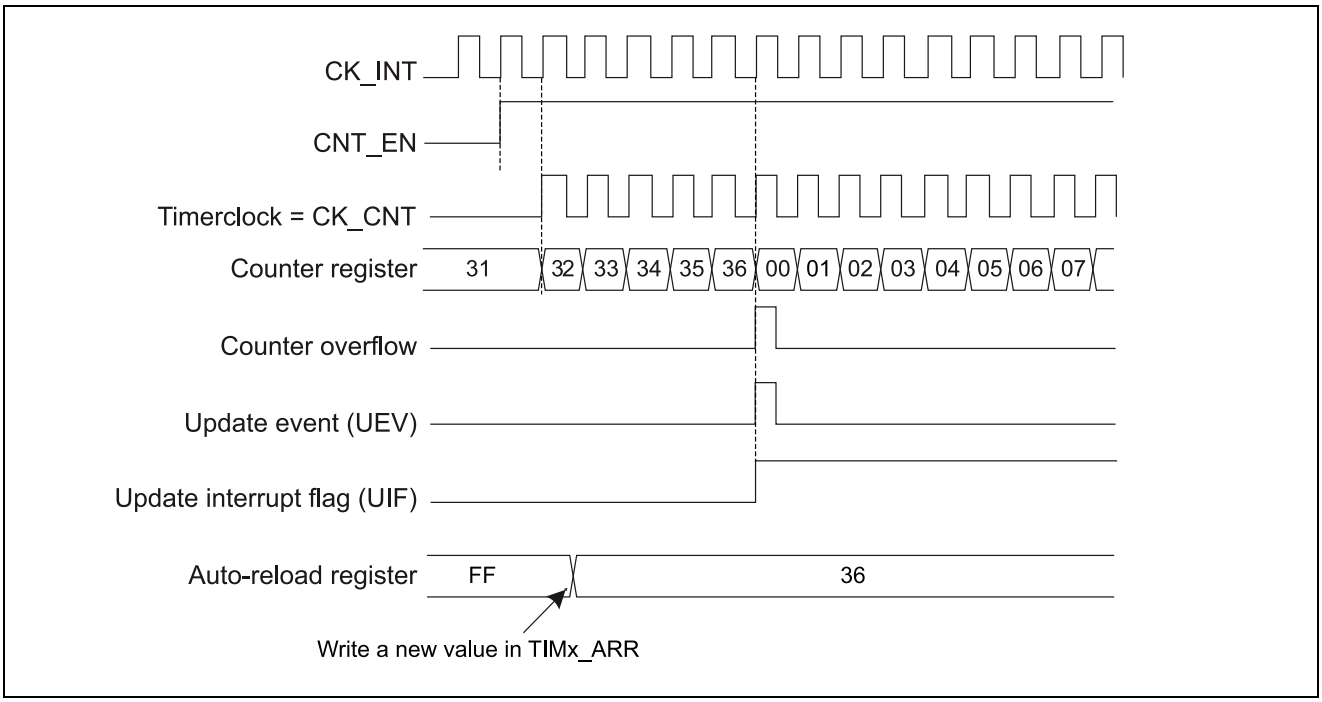
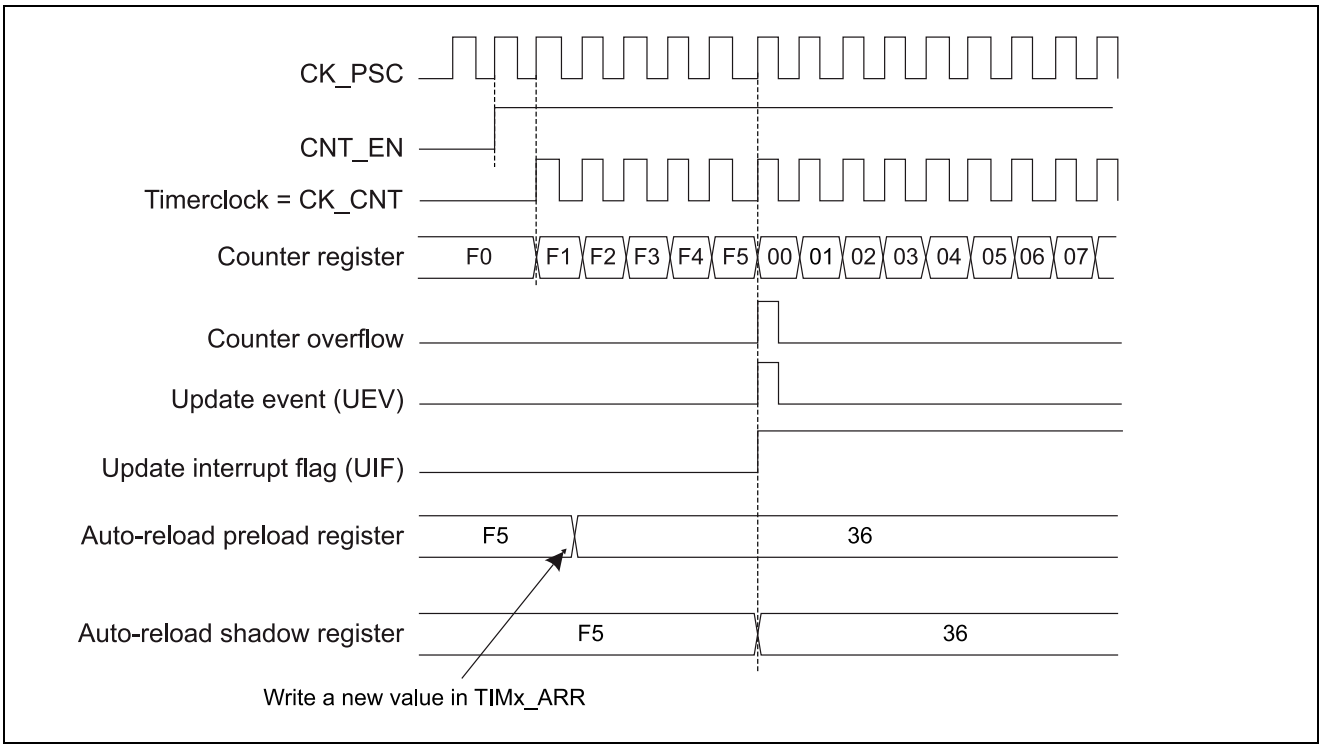


Fig 15.3-8 Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generate at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

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

Fig 15.3-9 Counter timing diagram, internal clock divided by 1

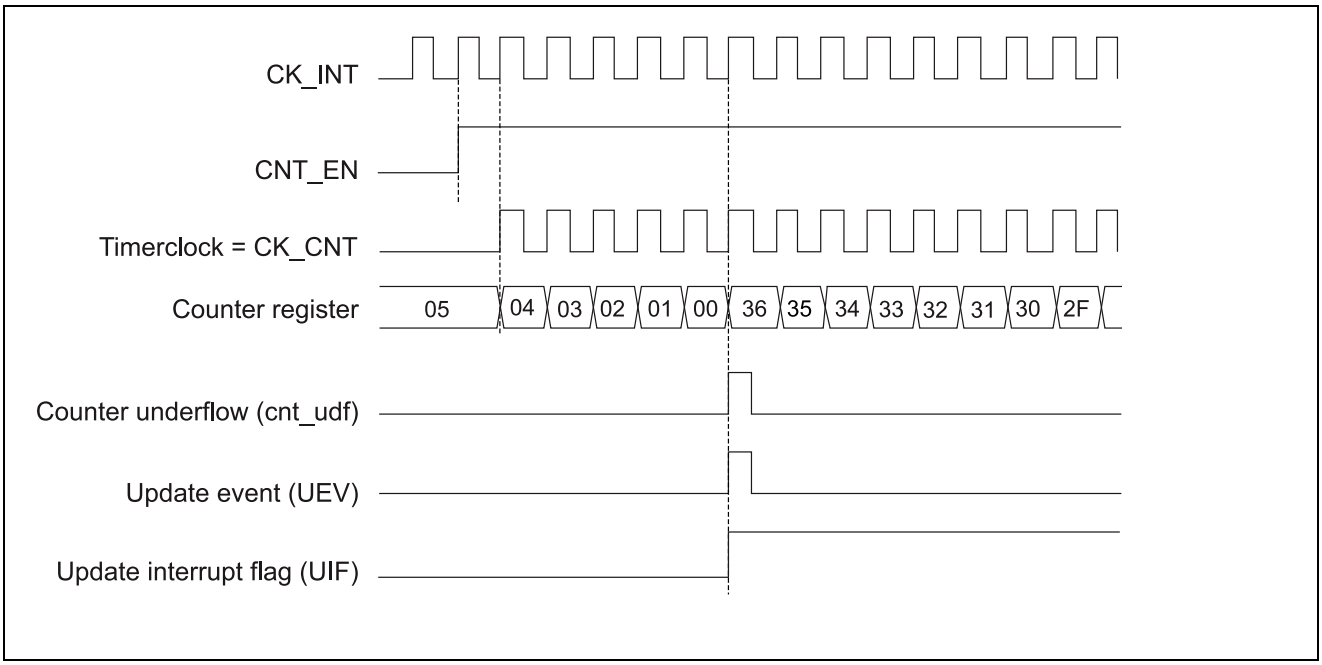


Fig 15.3-10 Counter timing diagram, internal clock divided by 2

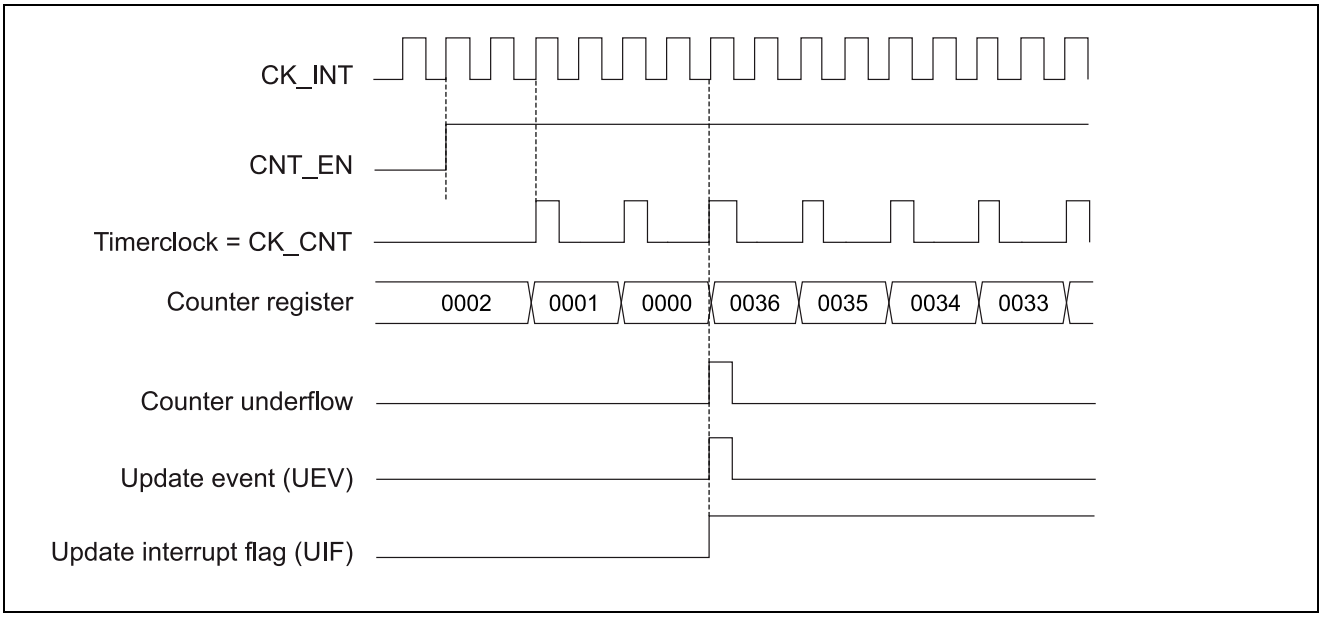


Fig 15.3-11 Counter timing diagram, internal clock divided by 4

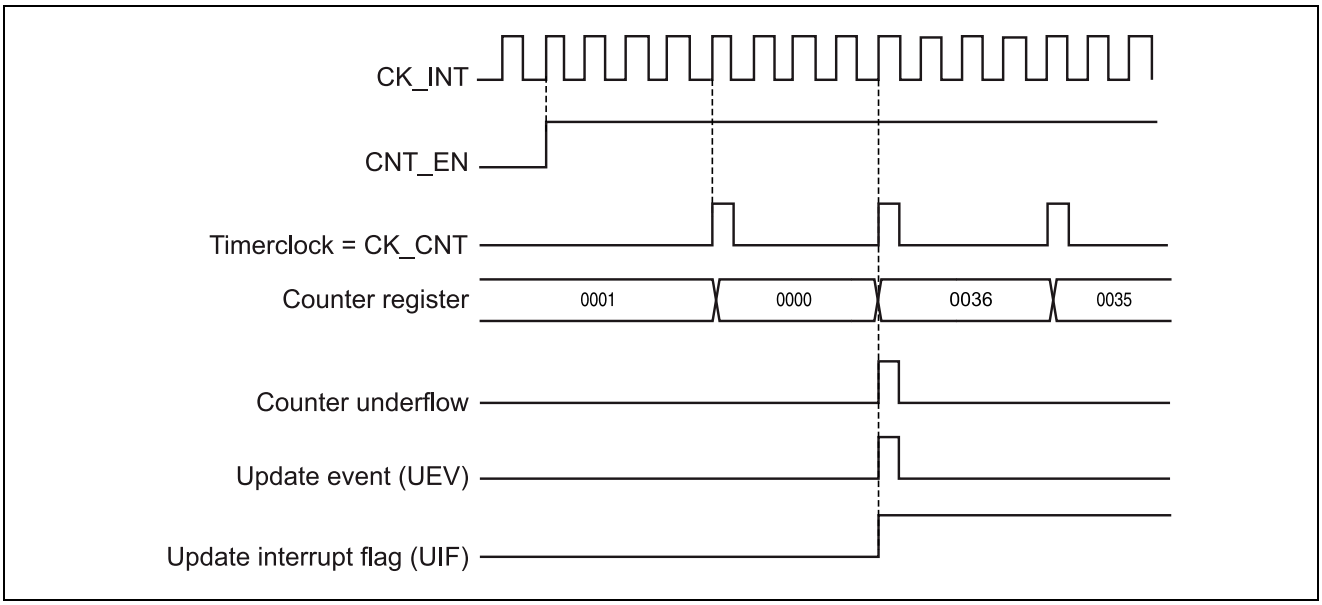


Fig 15.3-12 Counter timing diagram, internal clock divided by N

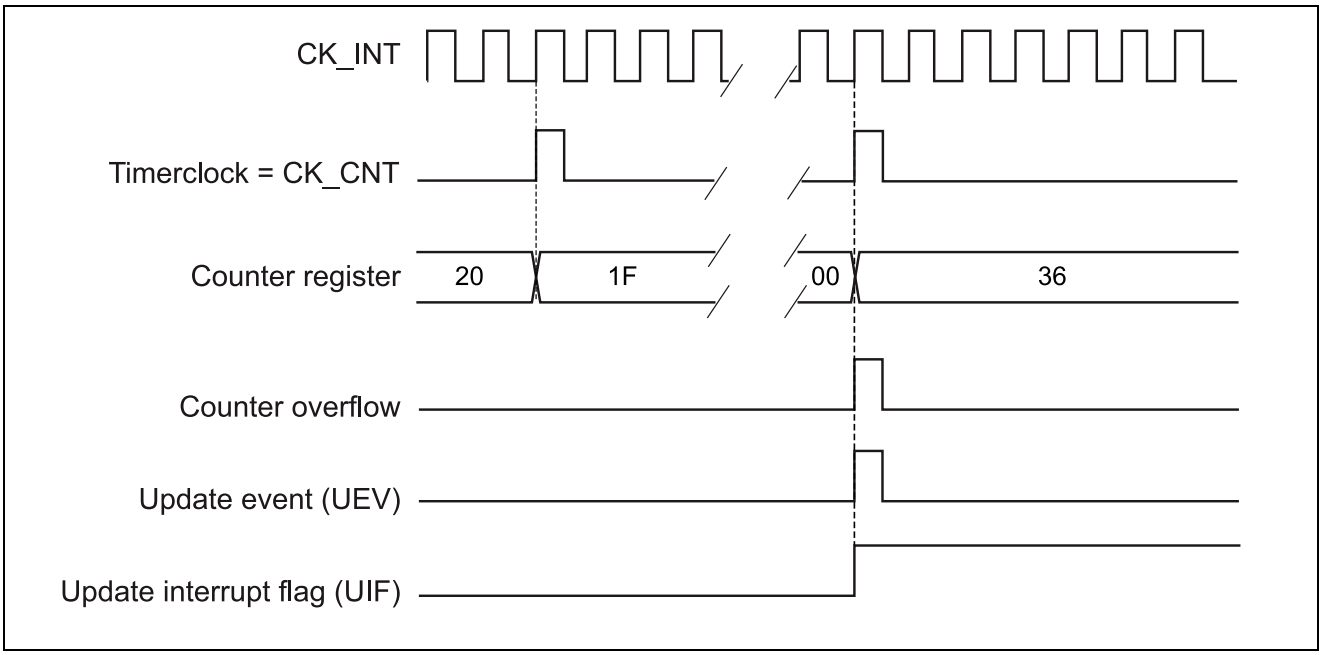
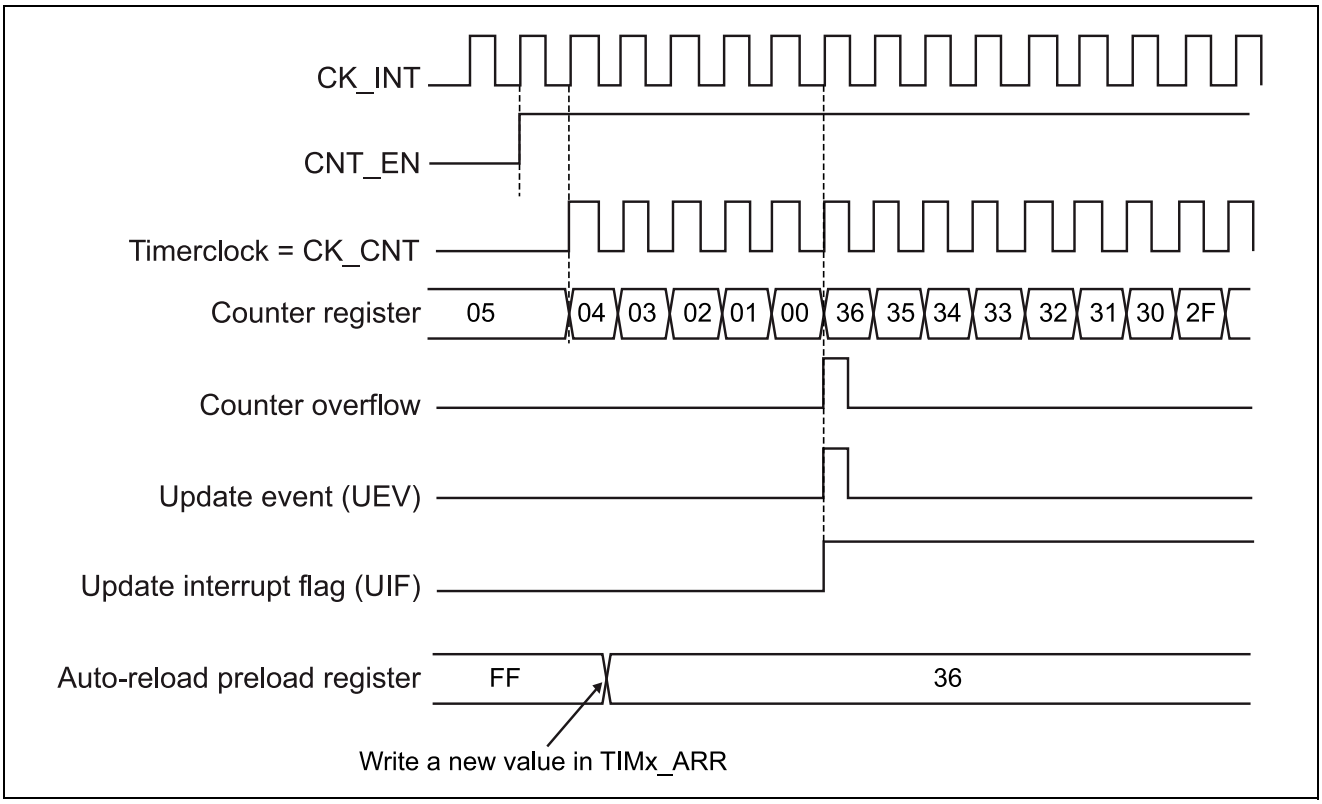


Fig 15.3-13 Counter timing diagram, Update event



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) –1, generates a counter overflow event, then counts from the auto- reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto- reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

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

Fig 15.3-14 Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6

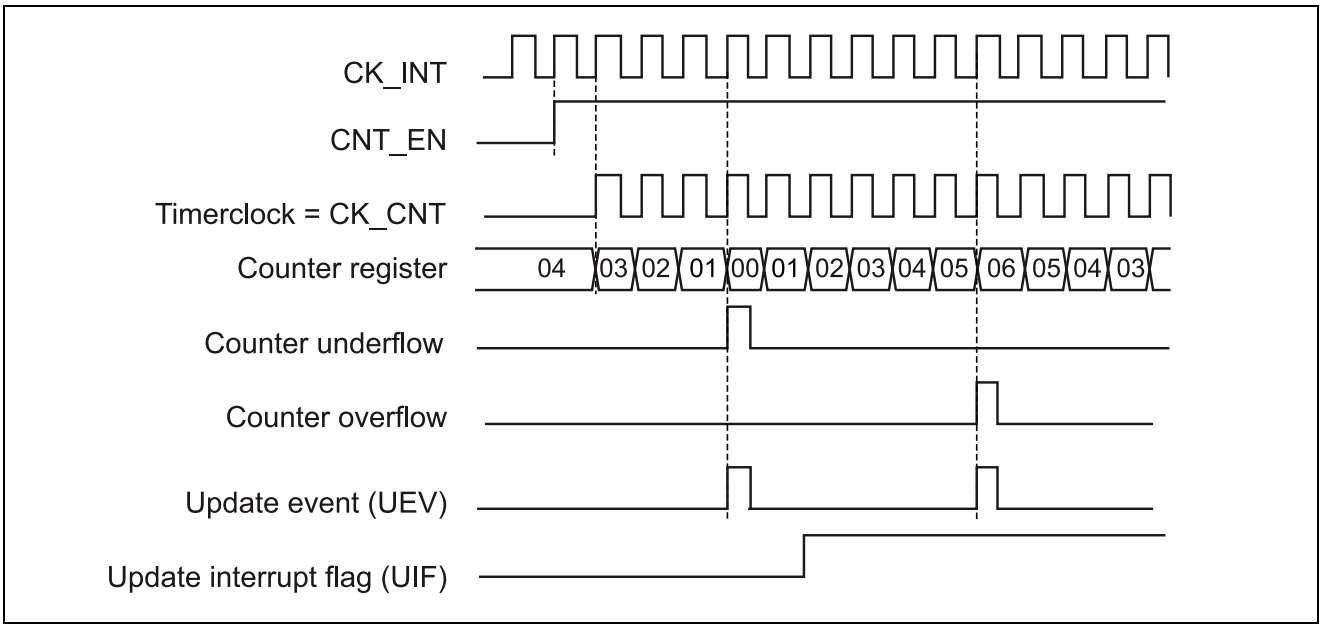


Fig 15.3-15 Counter timing diagram, internal clock divided by 2

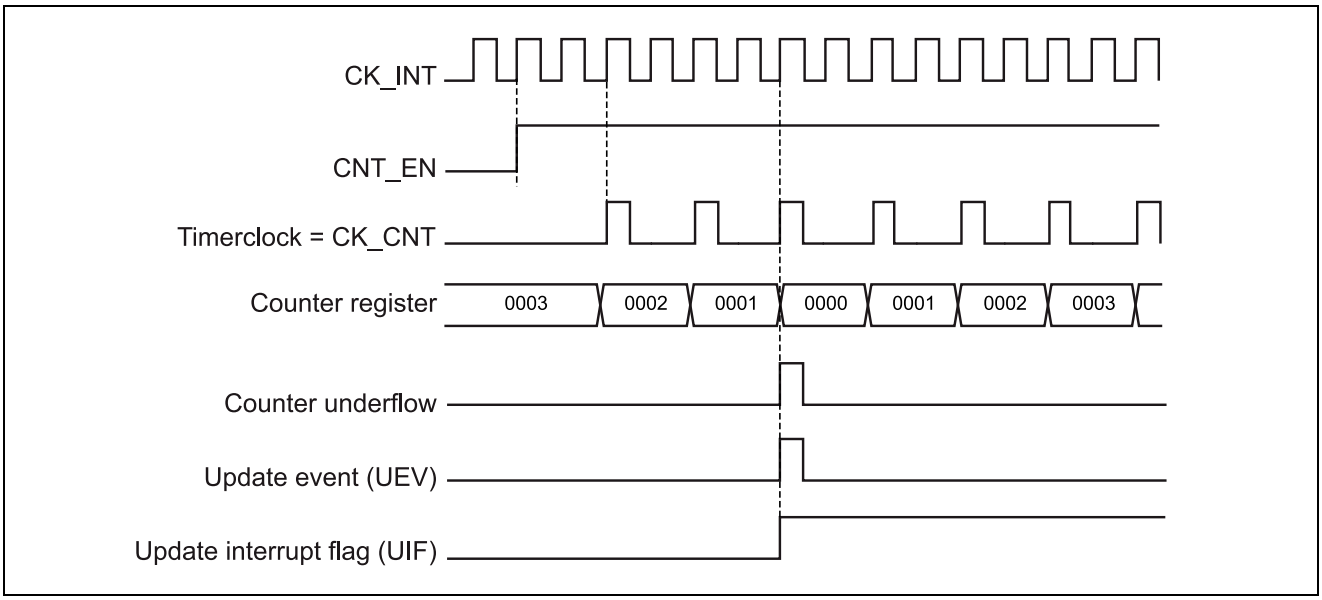


Fig 15.3-16 Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

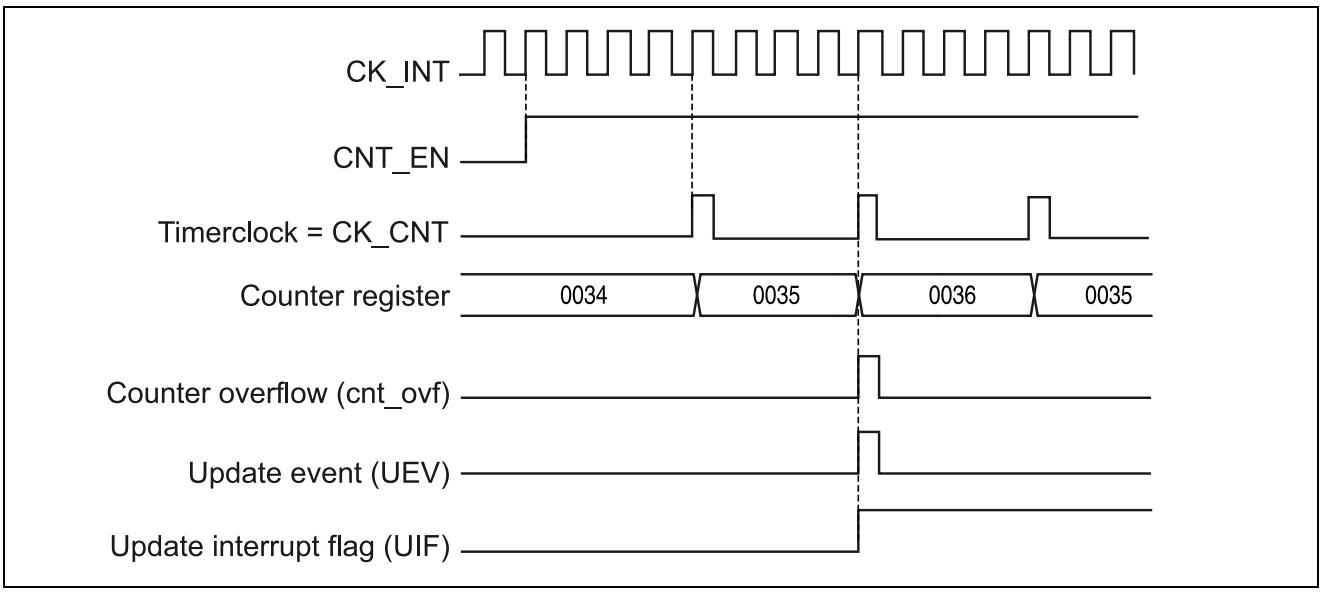


Fig 15.3-17 Counter timing diagram, internal clock divided by N

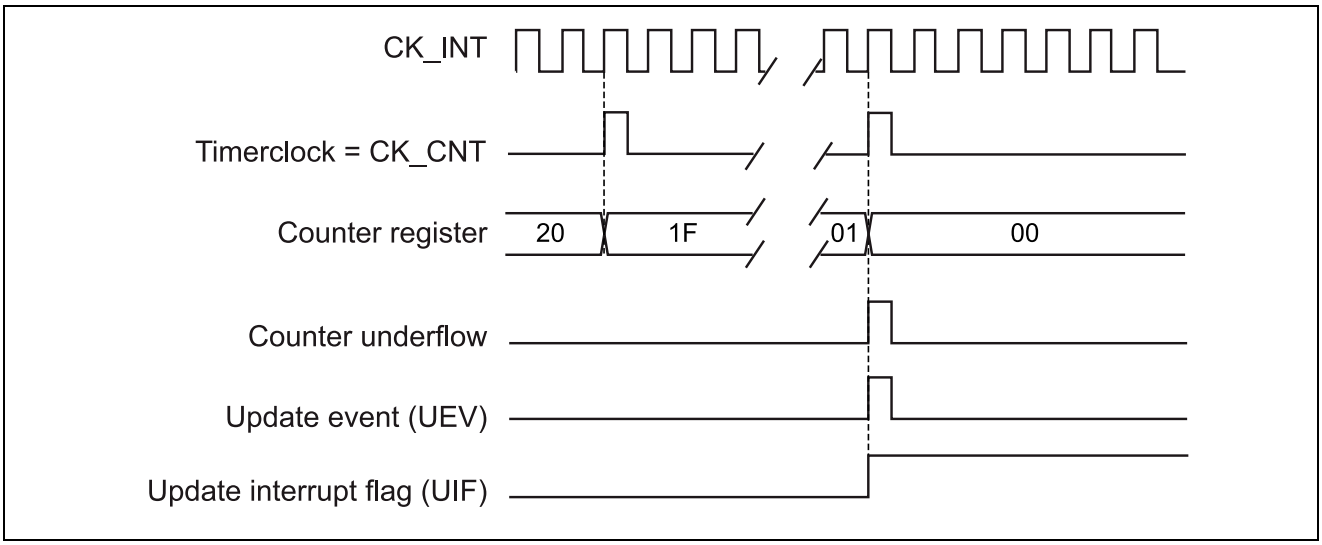


Fig 15.3-18 Counter timing diagram, Update event with ARPE=1 (counter underflow)

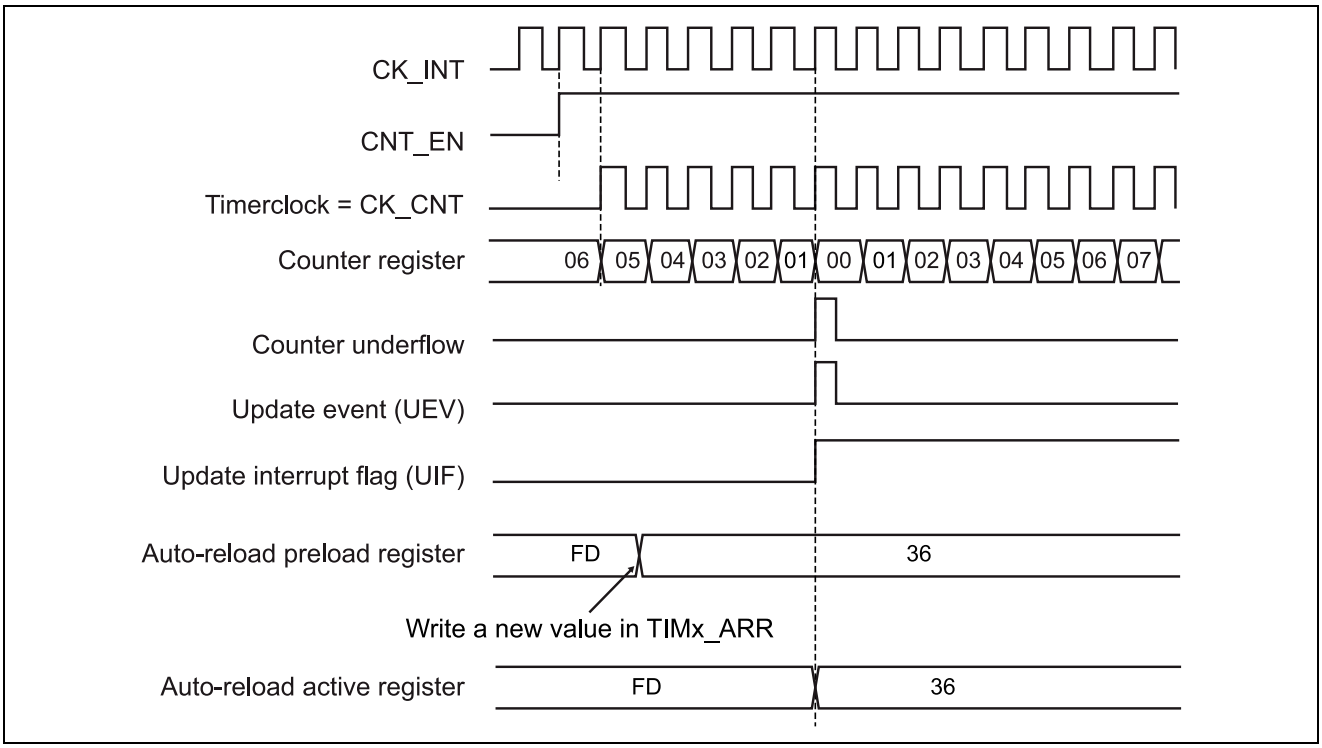
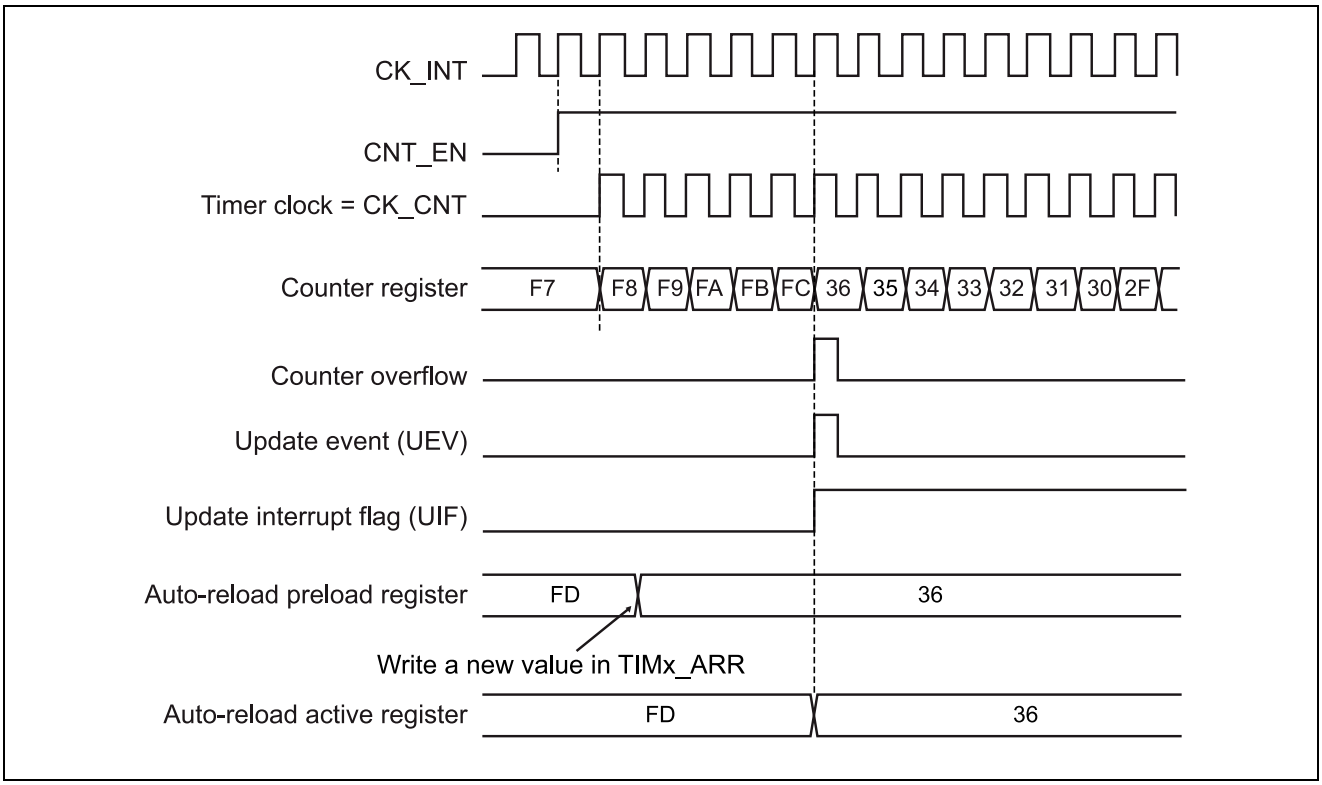


Fig 15.3-19 Counter timing diagram, Update event with ARPE=1 (counter overflow)



15.3.3 Clock selection

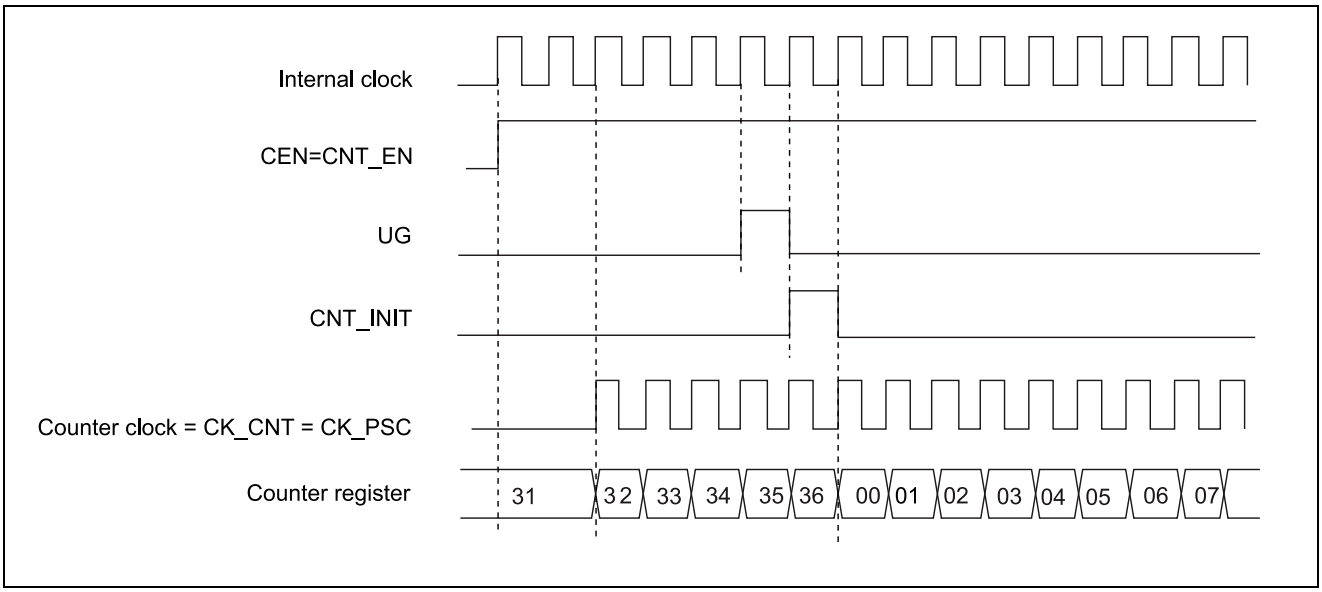
The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR).
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer1 can be configured to act as a prescaler for Timer 2. Refer to section 15.3.15 for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

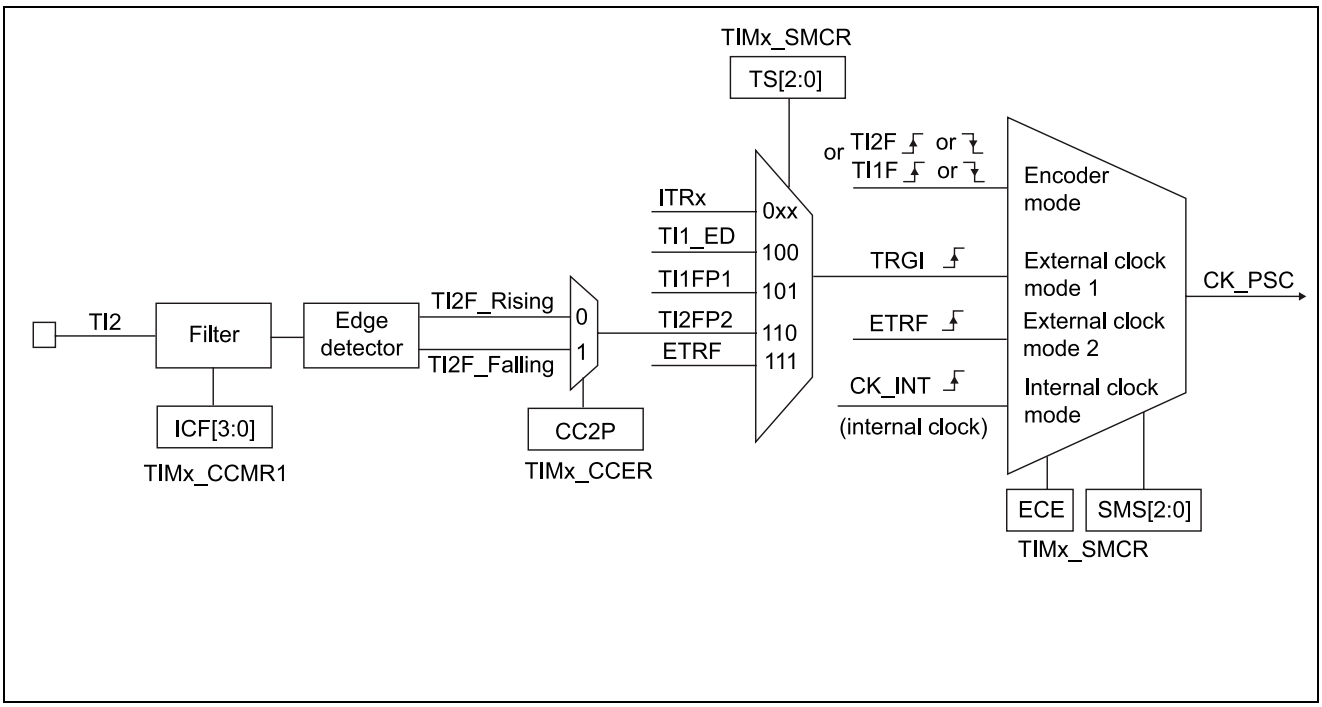
Fig 15.3-20 Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input

Fig 15.3-21 TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use

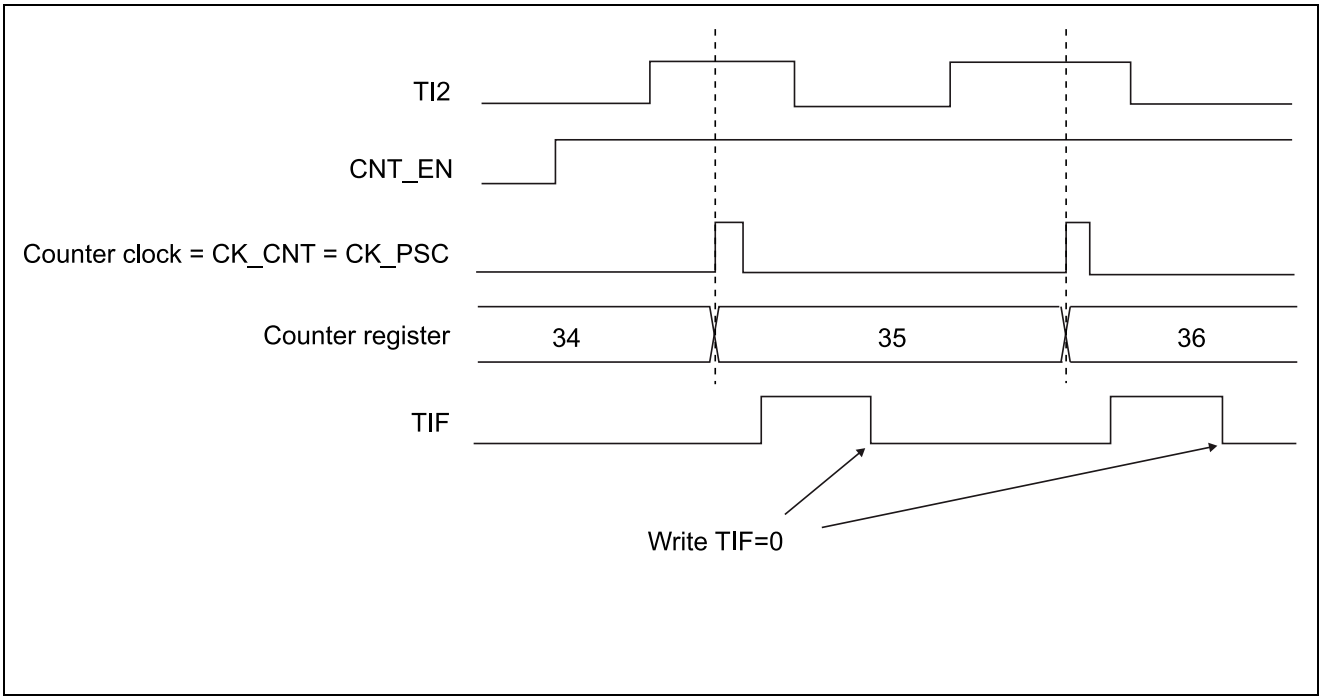
the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= ‘01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

   Note: *The capture prescaler is not used for triggering, so there’s no need to configure it.*
3. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set. The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

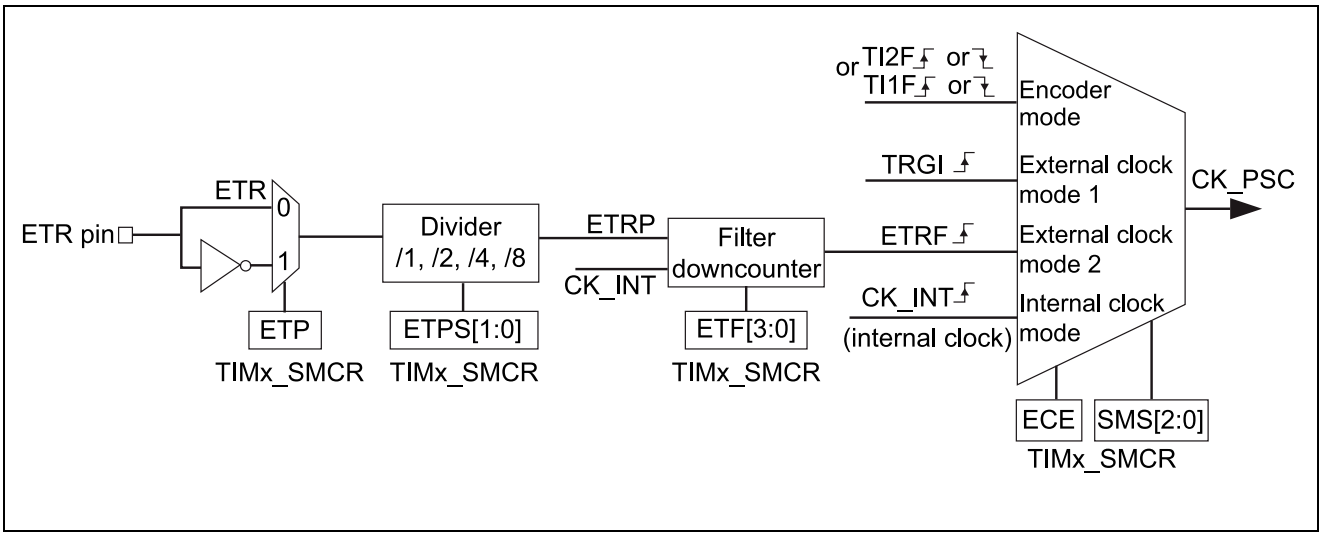
Fig 15.3-22 TI2 external clock connection example



External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register. The counter can count at each rising or falling edge on the external trigger input ETR.

Fig 15.3-23 External trigger input block



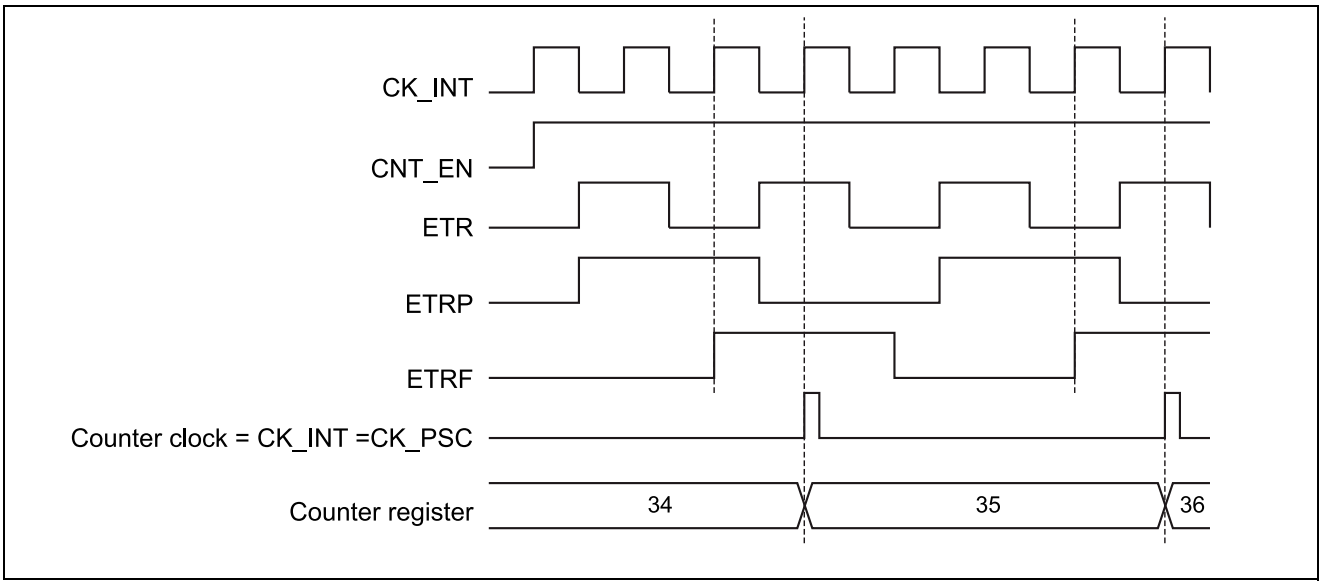
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

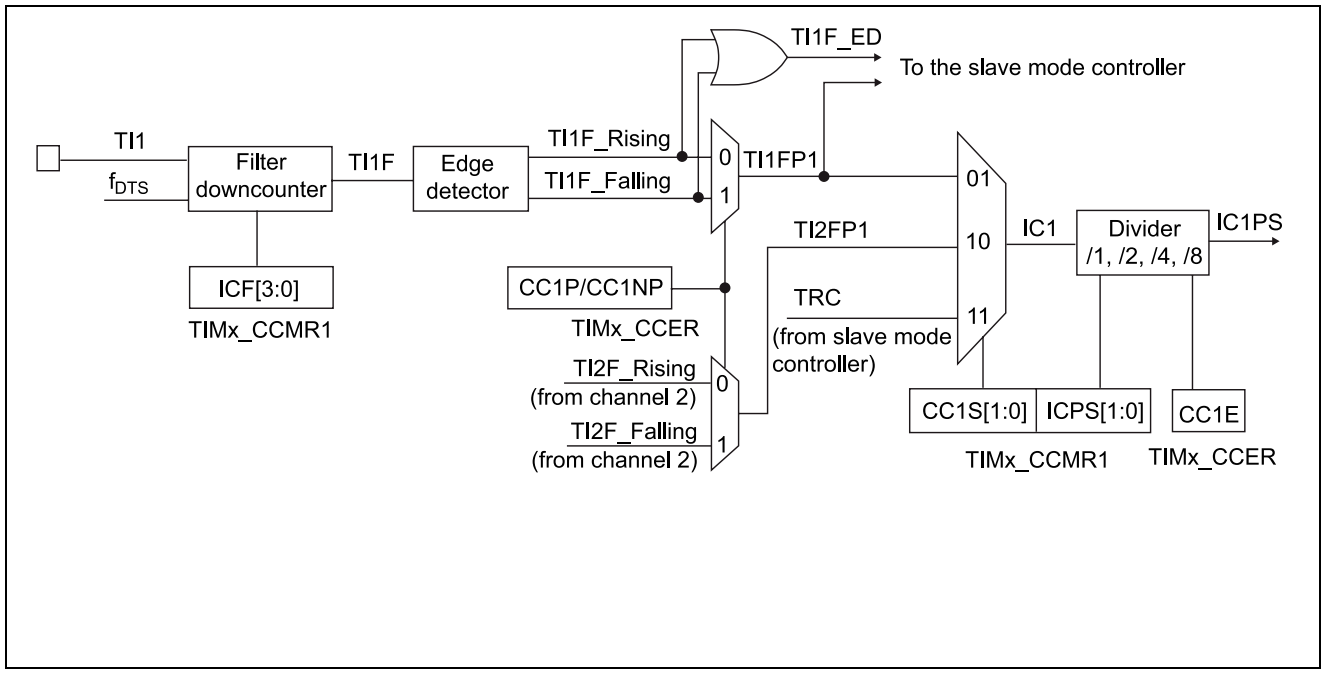
Fig 15.3-24 Control circuit in external clock mode 2



15.3.4 Capture/compare channels

Each Capture/Compare channel (see Figure 15.3-25) is built around a capture/compare register (including a shadow register), an 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).

Fig 15.3-25 Capture/compare channel (example: channel 1 input stage)



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.

Fig 15.3-26 Capture/compare channel 1 main circuit

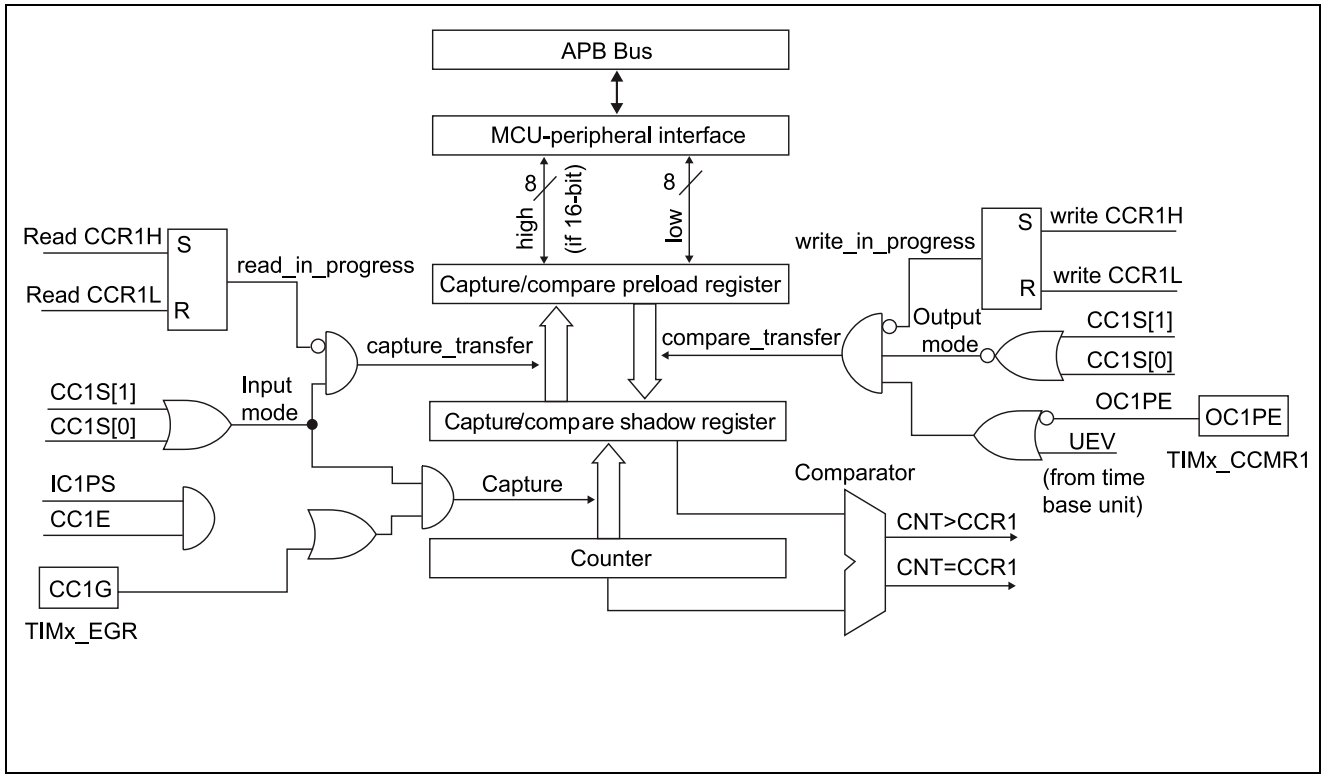
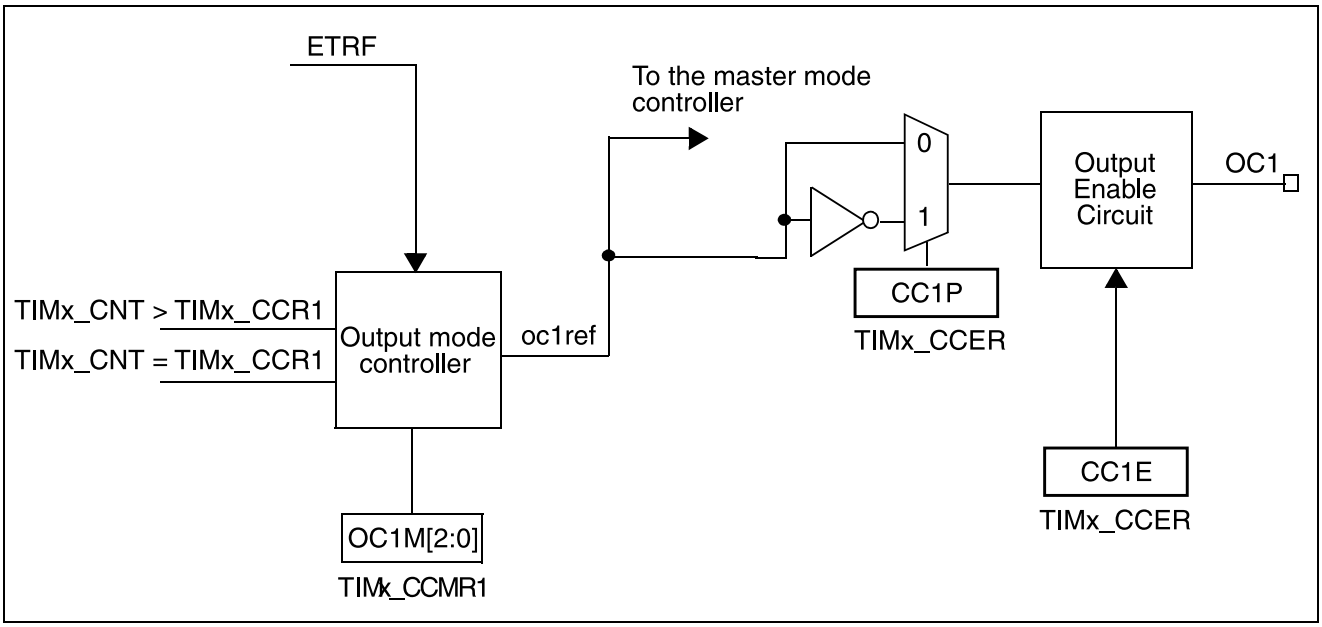


Fig 15.3-27 Output stage of capture/compare channel (channel 1)



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.

15.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written to 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- 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 and the TIMx_CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the TIx inputs). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock

cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on TI1 when eight consecutive samples with the new level have been detected (sampled at f DTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

- Select the edge of the active transition on the TI1 channel by writing the CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
- 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, and/or the DMA request by setting the CC1DE 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.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

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 and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.*

15.3.6 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

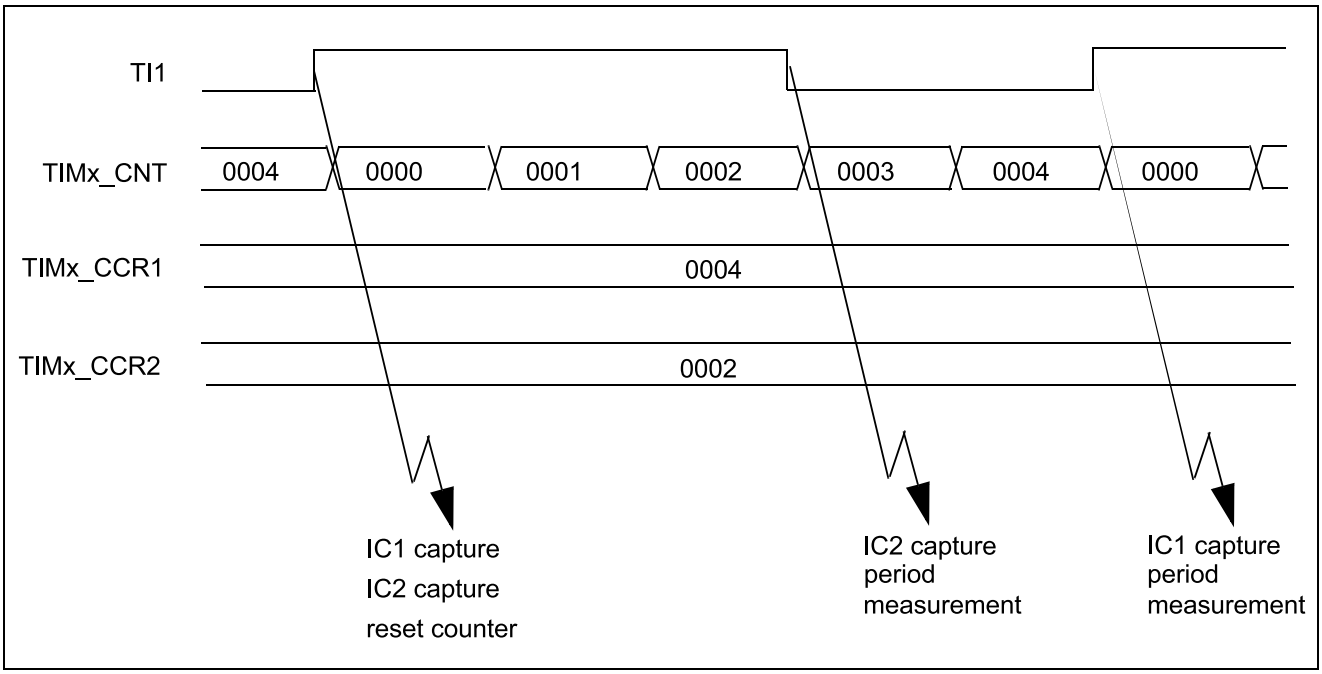
For example, the user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ‘0’ (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to ‘1’

(active on falling edge).

- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to ‘1 in the TIMx_CCER register.

Fig 15.3-28 PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

15.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

15.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

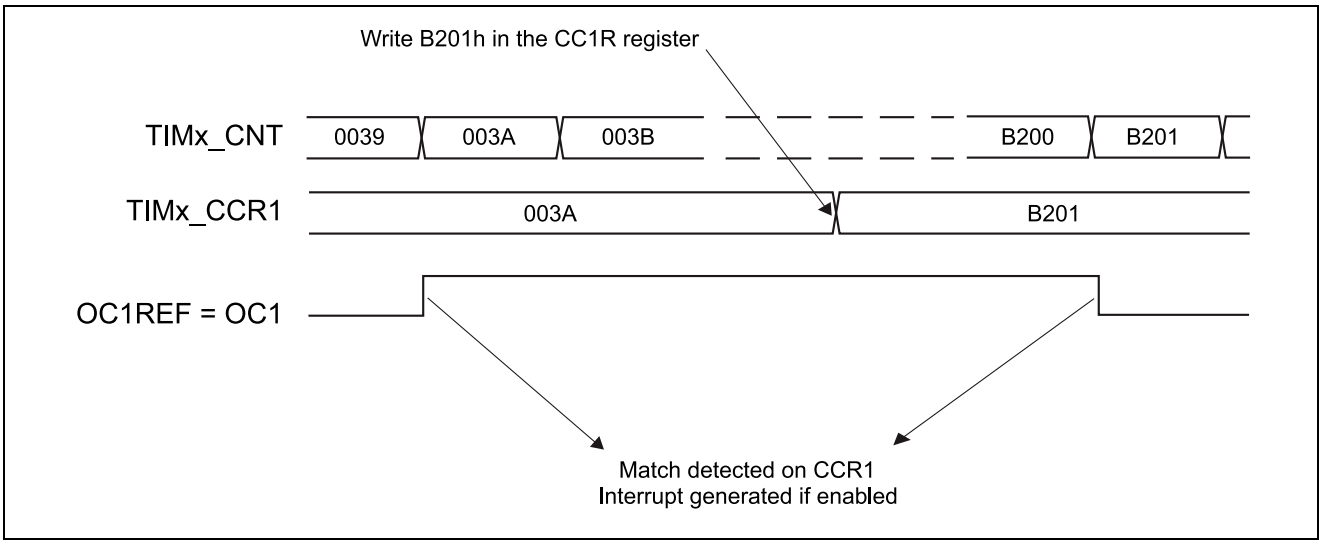
- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or 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).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register. In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode). Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, the user must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV).

Fig 15.3-29 Output compare mode, toggle on OC1



15.3.9 PWM mode

Pulse width modulation mode allows generating a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The user must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user has to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx $\leqslant$ TIMx_CNT or TIMx_CNT $\leqslant$ TIMx_CCRx (depending on the direction of the counter). However, to comply with the ETRF (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM= ‘000) to one of the PWM modes (OCxM= ‘110 or ‘111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

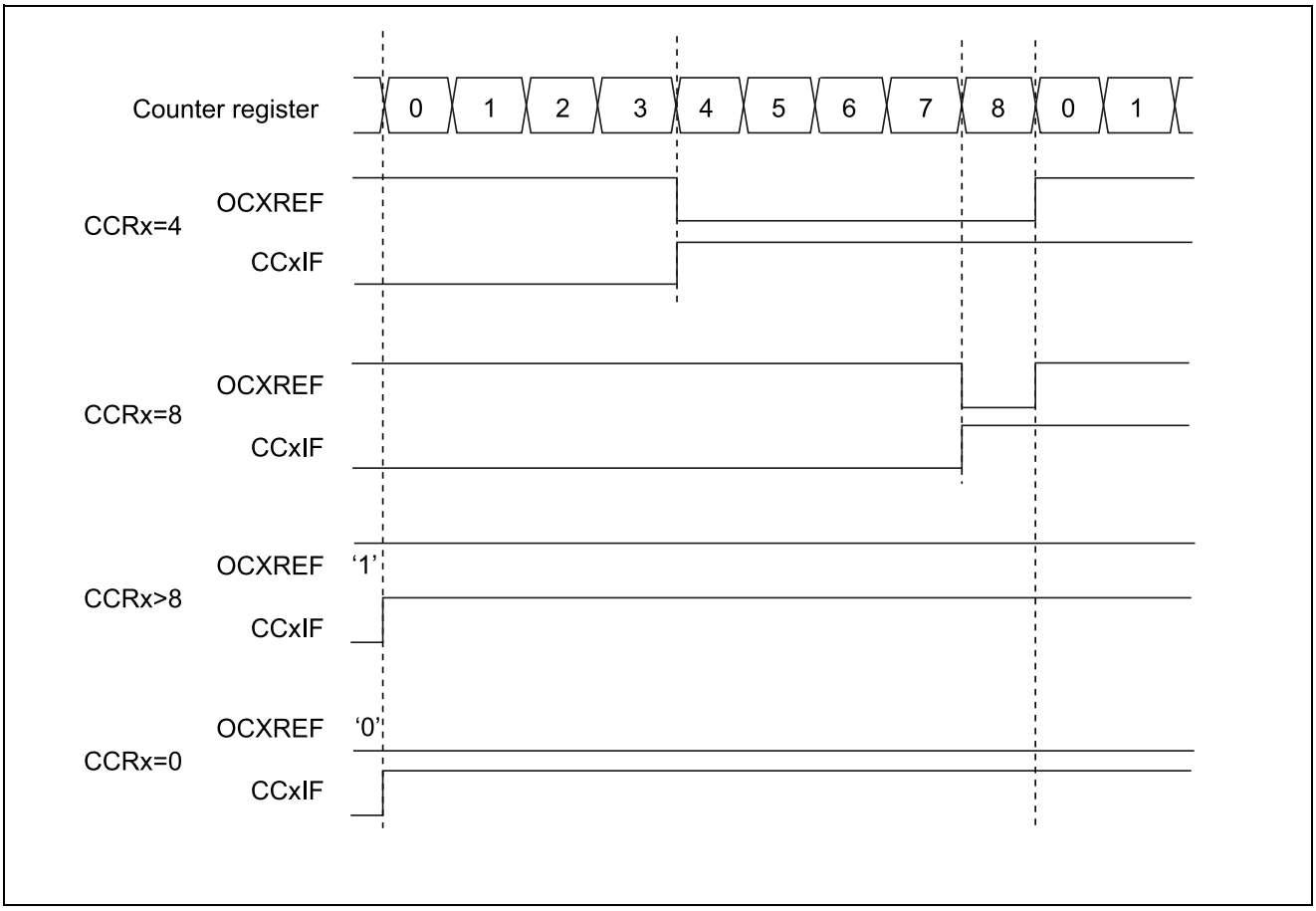
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1. If the compare value is 0 then OCxREF is held at ‘0. Figure 130 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Fig 15.3-30 Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high.
In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at ‘1. 0% PWM is not possible in this mode.

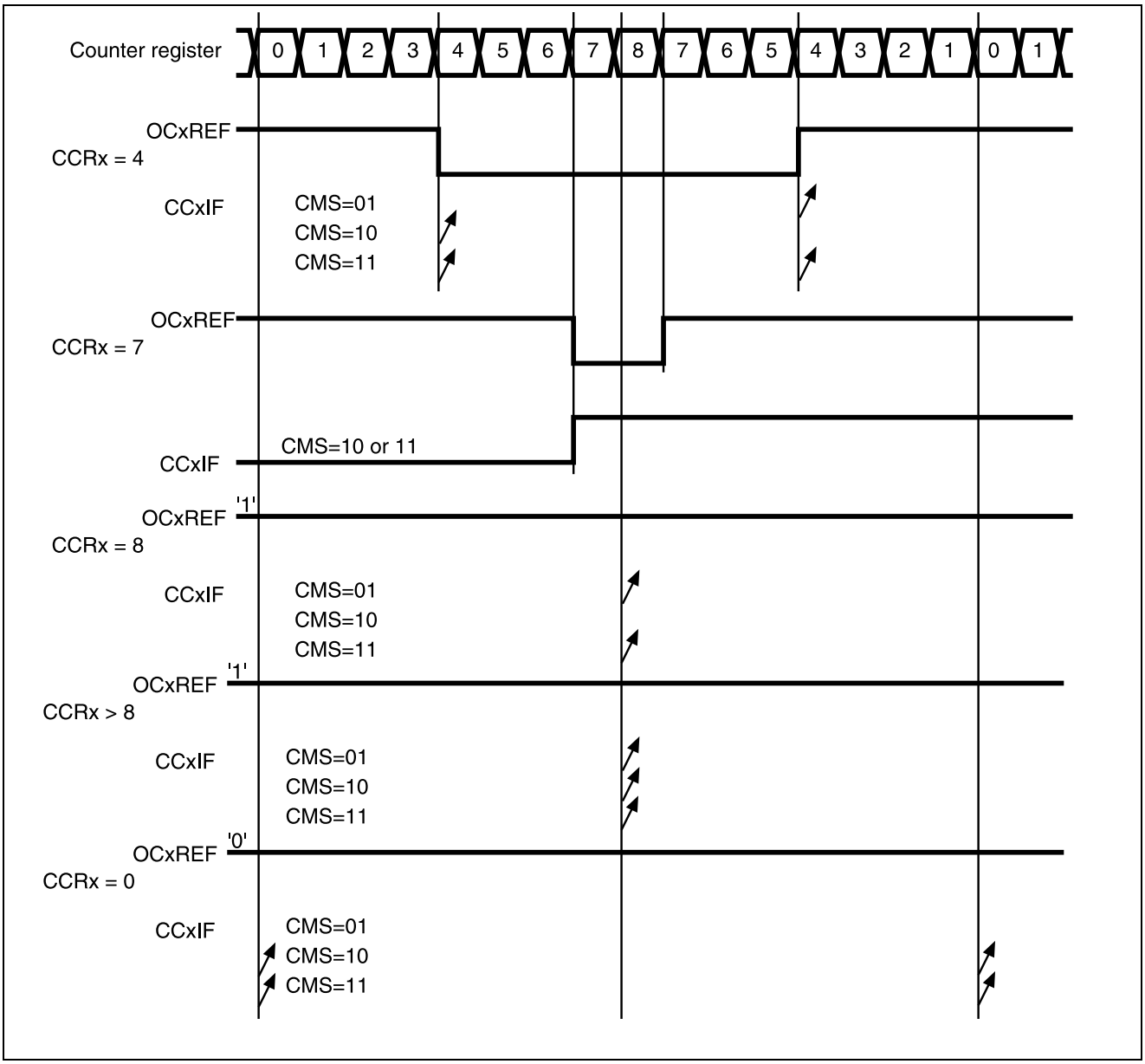
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software.

shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Fig 15.3-31 Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if the user writes a value in the counter that is greater than the auto-reload value (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if the user writes 0 or write the TIMx_ARR value in the counter but

no Update Event UEV is generated.

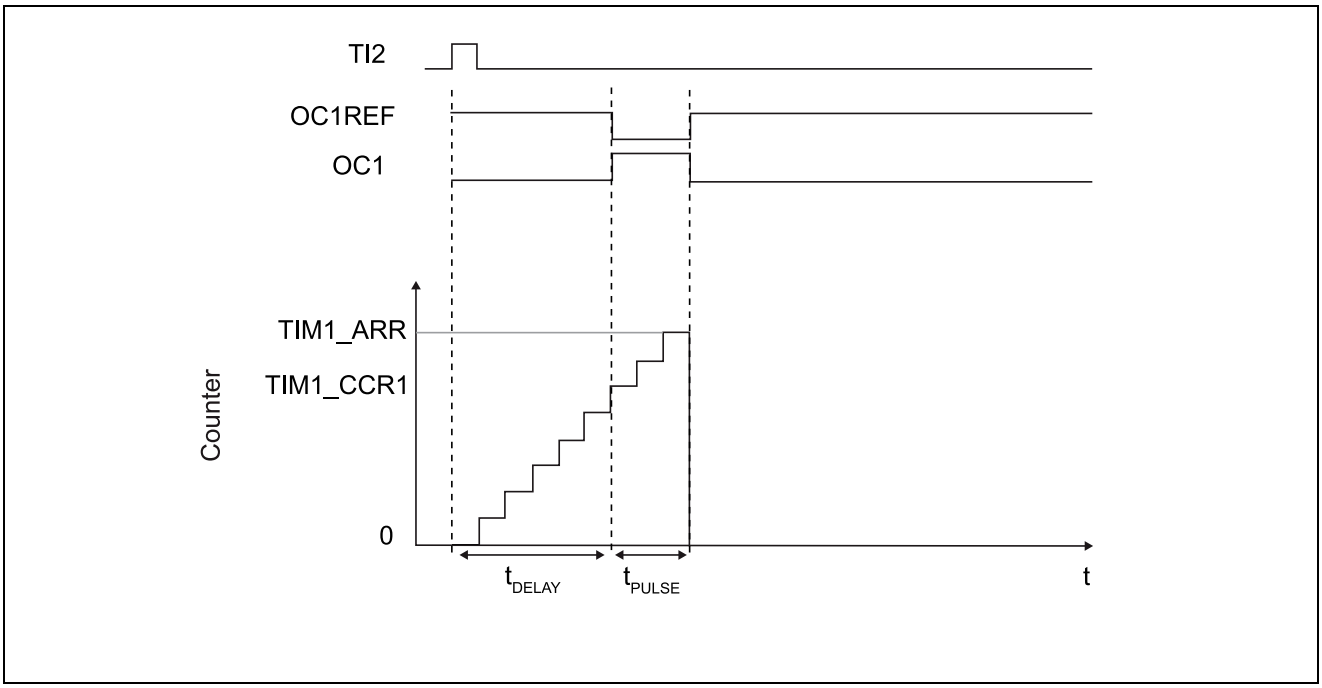
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

15.3.10 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.
Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.
A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≦ ARR (in particular, 0 < CCRx),
- In downcounting: CNT > CCRx

Fig 15.3-32 Example of one-pulse mode



For example the user may want to generate a positive pulse on OC1 with a length of t PULSE and after a delay of t DELAY as soon as a positive edge is detected on the TI2 input pin.
Let’s use TI2FP2 as trigger 1:

- Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.

- TI2FP2 must detect a rising edge, write CC2P=0 in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to ‘110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR + 1).
- Let us say user wants to build a waveform with a transition from ‘0 to ‘1 when a compare match occurs and a transition from ‘1 to ‘0 when the counter reaches the auto-reload value. To do this enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. The user can optionally enable the preload registers by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low. User only wants one pulse (Single mode), so write ’1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to ’0’, so the Repetitive Mode is selected.

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 t DELAY min we can get. To output a waveform with the minimum delay, the user can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

15.3.11 Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OCxCE enable bit of the corresponding TIMx_CCMRx register set to ’1’). The OCxREF signal remains Low until the next update event, UEV, occurs.
This function can only be used in output compare and PWM modes, and does not work in forced mode.
For example, the ETR signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

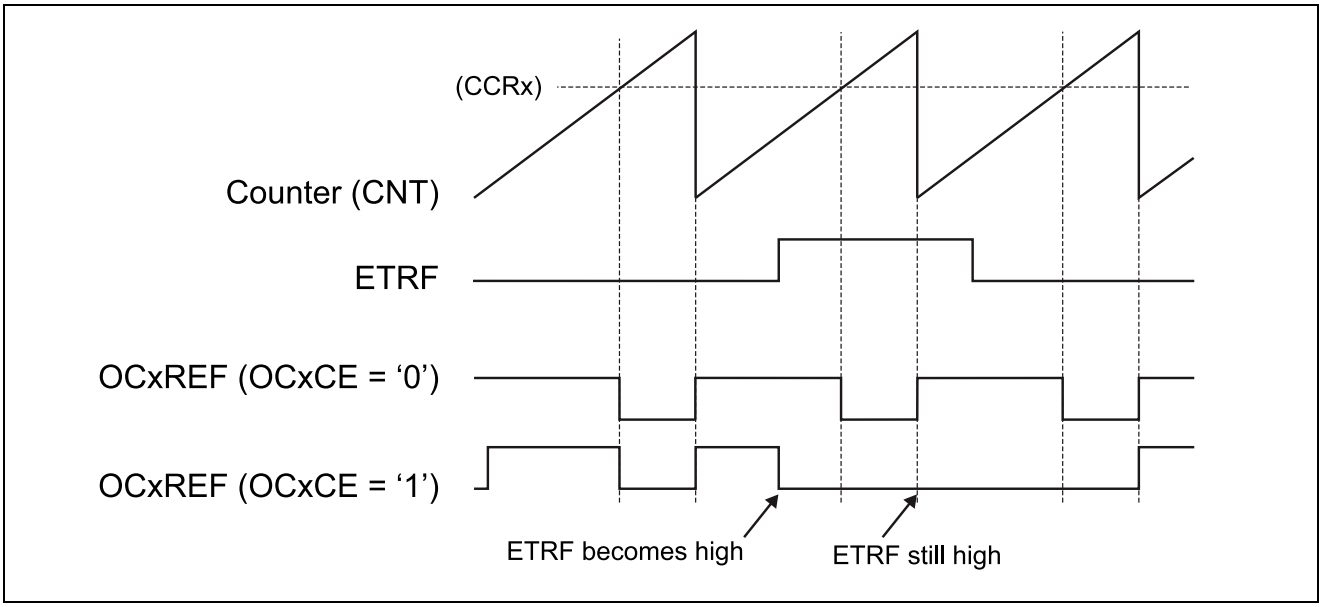
1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are

cleared to 00.

2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application’s needs.

Fig:15.3-33 shows the behavior of the OCxREF signal when the ETRF input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Fig 15.3-33 Clearing TIMx OCxREF



15.3.12 Encoder interface mode

To select Encoder Interface mode write SMS= ‘001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to Table 15.3-1. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0

to ARR or ARR down to 0 depending on the direction). So the user must configure TIMx_ARR before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don’t switch at the same time.

Tab 15.3-1 Counting direction versus encoder signals

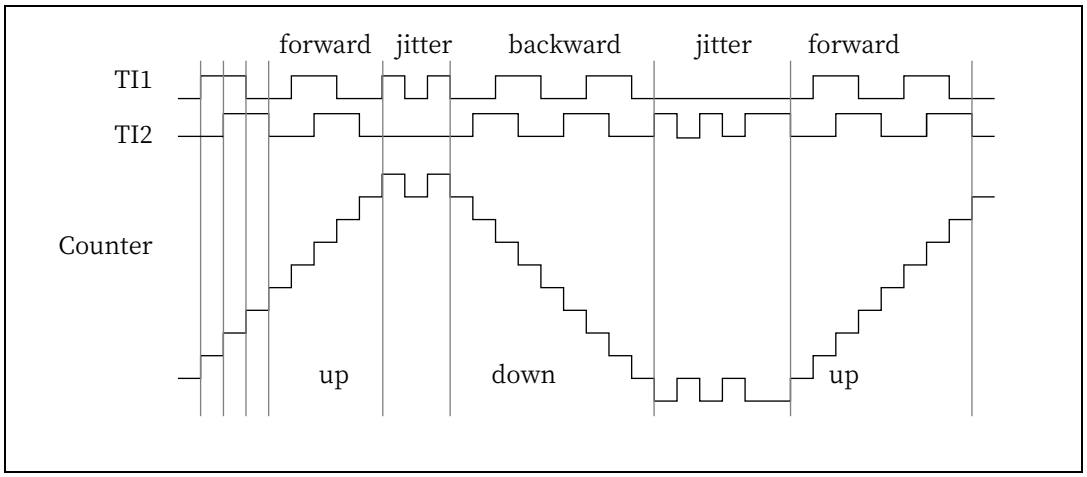
Active edge	**Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)**	**TI1FP1 signal**		**TI2FP2 signal**	
		Rising	Falling	Rising	Falling
Counting onTI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Fig：15.3-34gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

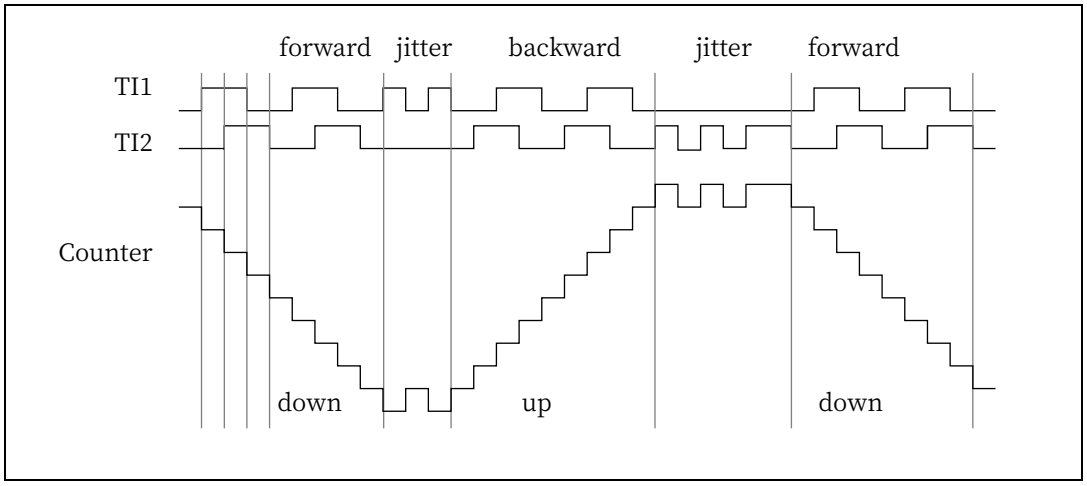
- CC1S= ‘01’(TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= ‘01’(TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P= ‘0’, CC1NP = ‘0’, IC1F =’0000’(TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P= ‘0’, CC2NP = ‘0’, IC2F =’0000’(TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= ‘011’(TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN = 1 (TIMx_CR1 register, Counter is enabled)

Fig 15.3-34 Example of counter operation in encoder interface mode



Fig：15.3-35gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Fig 15.3-35 Example of encoder interface mode with TI1FP1 polarity inverted



The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. The user can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. The user can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

15.3.13 Timer input XOR function

The TI1S bit in the TIM1_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3. The XOR output can be used

with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in ***14.3.17***

15.3.14 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

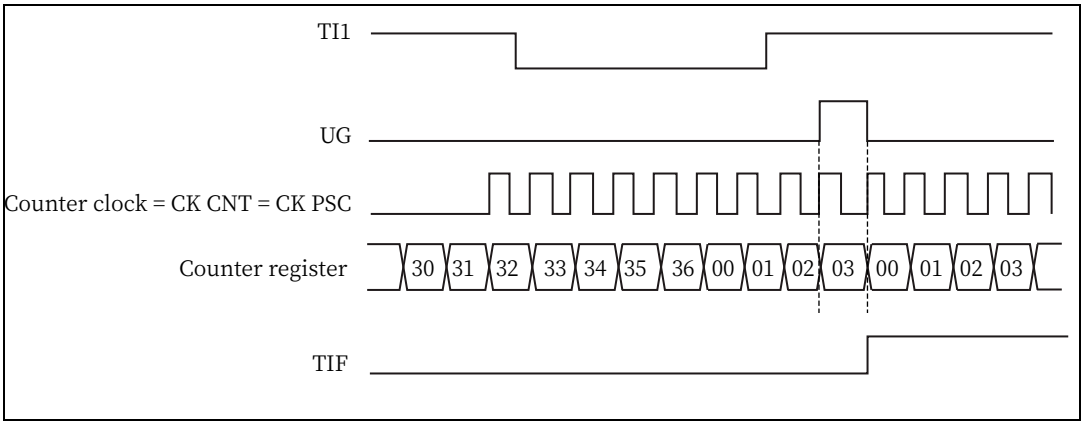
The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated. In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don’t need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so the user does not need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

Fig: shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Fig 15.3-36 Control circuit in reset mode



Slave mode: Gated mode

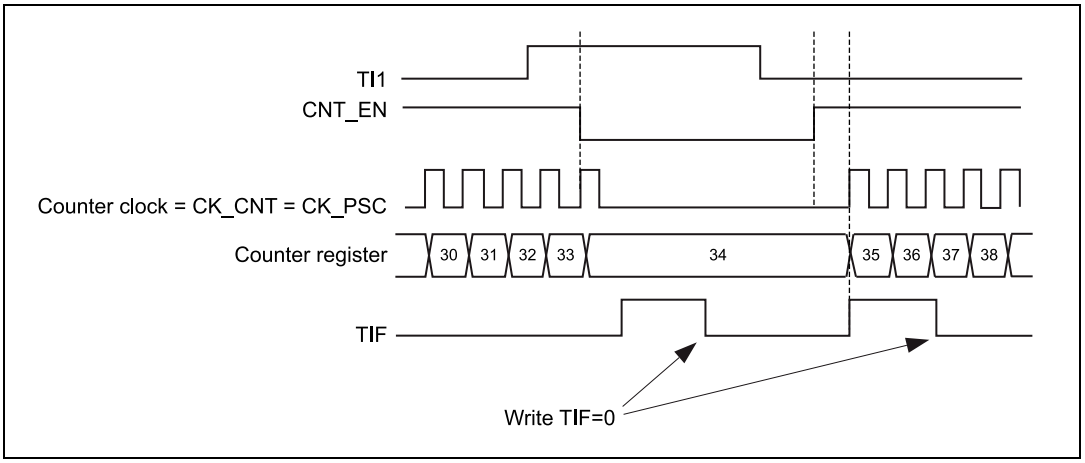
The counter can be enabled depending on the level of a selected input. In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so the user does not need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Fig 15.3-37 Control circuit in gated mode



15.3.14.1 Slave mode: Trigger mode

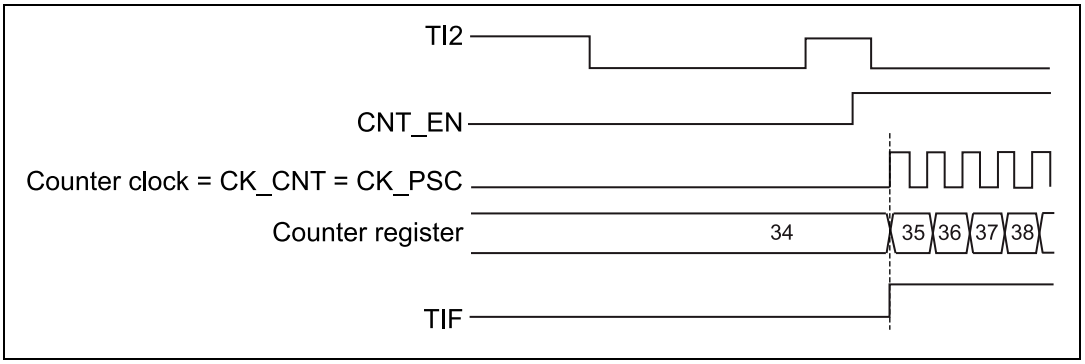
The counter can start in response to an event on a selected input. In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so the user does not need to configure it. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Fig 15.3-38 Control circuit in trigger mode



Slave mode: External Clock mode 2 + trigger mode

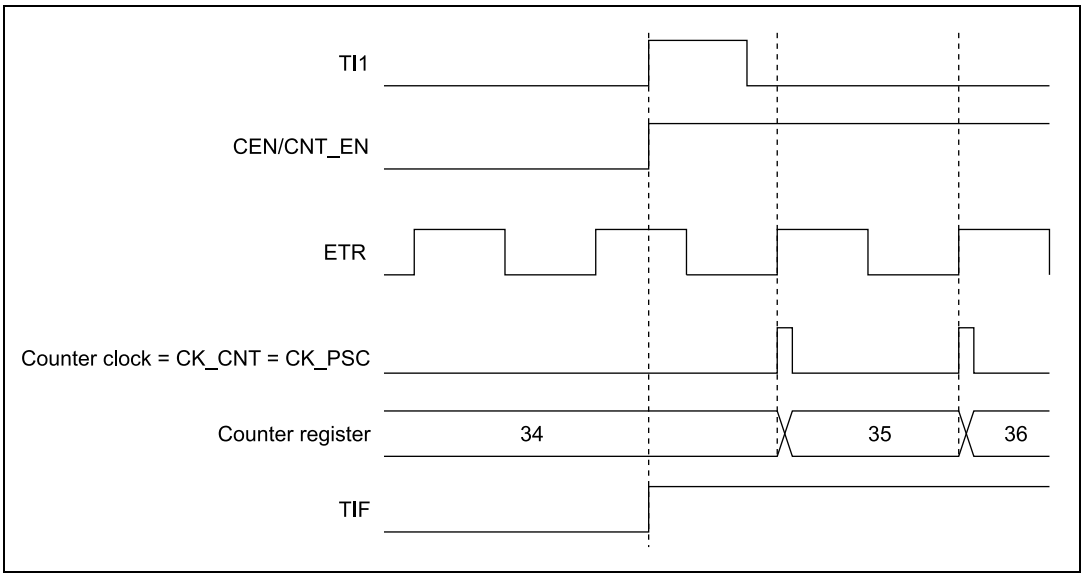
The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01 in TIMx_CCMR1 register to select only the input capture source
   - CC1P = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.
The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Fig 15.3-39 Control circuit in external clock mode 2 + trigger mode



15.3.15 Timer synchronization

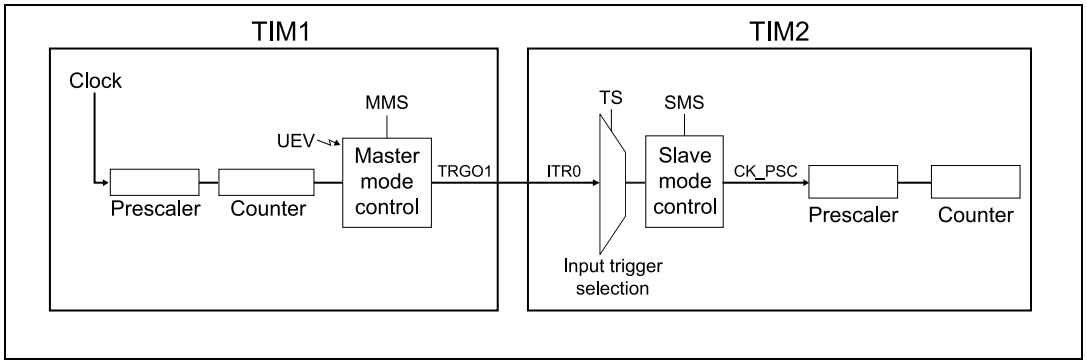
The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 15.3-40 presents an overview of the trigger selection and the master mode selection blocks

Note: *The clock of the slave timer must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Using one timer as prescaler for another timer

Fig 15.3-40 Master/Slave timer example



For example, the user can configure Timer 1 to act as a prescaler for Timer 2 (see Figure 15.3-40). To do this:

- Configure Timer 1 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS=010 in the TIM1_CR2 register, a rising edge is output on TRGO1 each time an update event is generated.

- To connect the TRGO1 output of Timer 1 to Timer 2, Timer 2 must be configured in slave mode using ITR0 as internal trigger. You select this through the TS bits in the TIM2_SMCR register (writing TS=000).
- Then you put the slave mode controller in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes Timer 2 to be clocked by the rising edge of the periodic Timer 1 trigger signal (which correspond to the timer 1 counter overflow).
- Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: *If OCx is selected on Timer 1 as trigger output (MMS=1xx), its rising edge is used to clock the counter of timer 2*

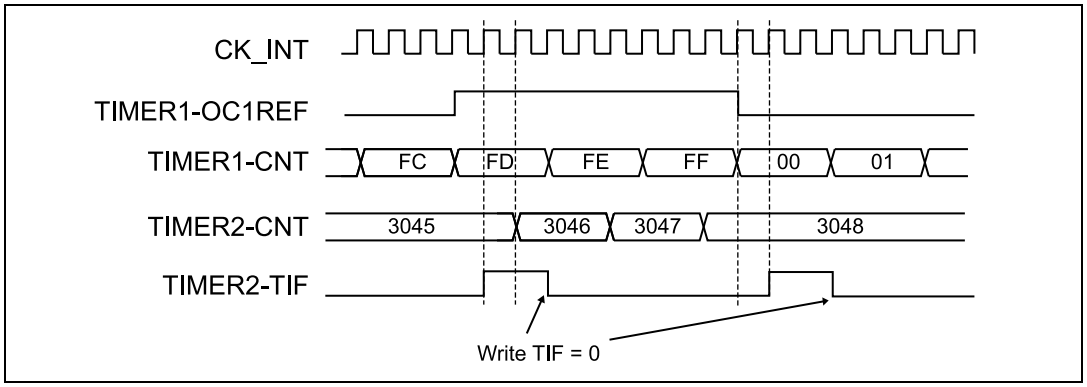
Using one timer to enable another timer

In this example, we control the enable of Timer 2 with the output compare 1 of Timer 1. Refer to Figure 15.3-40 for connections. Timer 2 counts on the divided internal clock only when OC1REF of Timer 1 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_CNT}$ /3).

- Configure Timer 1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in gated mode (SMS=101 in TIM2_SMCR register).
- Enable Timer 2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
- Start Timer 1 by writing ‘1 in the CEN bit (TIM1_CR1 register).

Note: *The counter 2 clock is not synchronized with counter 1, this mode only affects the Timer 2 counter enable signal.*

Fig 15.3-41 Gating timer 2 with OC1REF of timer 1



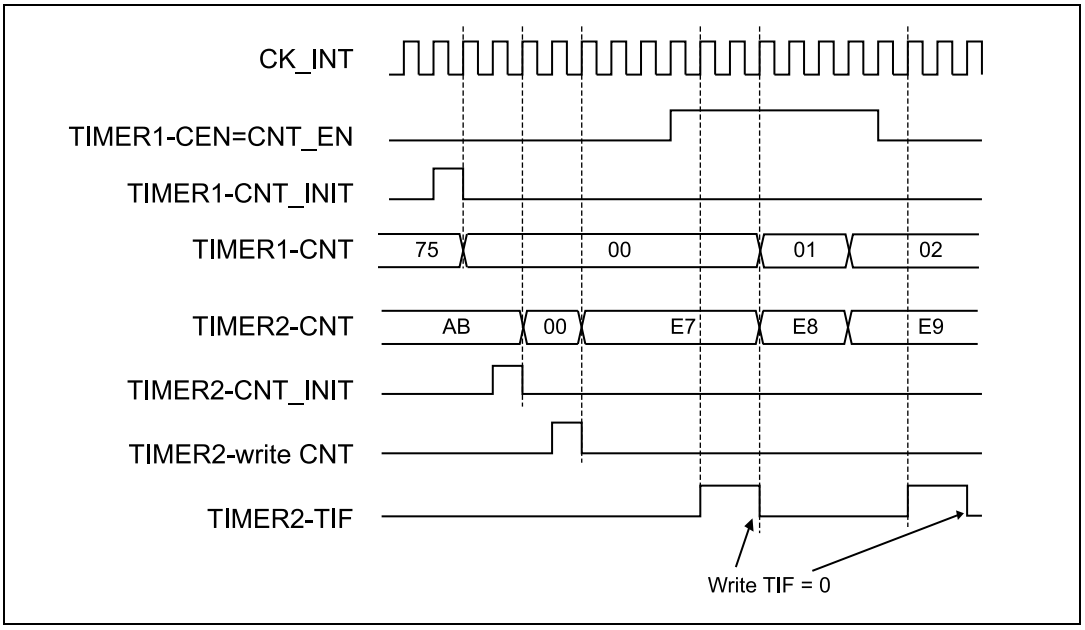
In the example in Figure 15.3-41, the Timer 2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting Timer 1. You can then write any value you want in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example, we synchronize Timer 1 and Timer 2. Timer 1 is the master and starts from 0. Timer 2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer 2 stops when Timer 1 is disabled by writing ‘0 to the CEN bit in the TIM1_CR1 register:

- Configure Timer 1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).

- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in gated mode (SMS=101 in TIM2_SMCR register).
- Reset Timer 1 by writing ‘1 in UG bit (TIM1_EGR register).
- Reset Timer 2 by writing ‘1 in UG bit (TIM2_EGR register).
- Initialize Timer 2 to 0xE7 by writing ‘0xE7’ in the timer 2 counter (TIM2_CNTL).
- Enable Timer 2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
- Start Timer 1 by writing ‘1 in the CEN bit (TIM1_CR1 register).
- Stop Timer 1 by writing ‘0 in the CEN bit (TIM1_CR1 register).

Fig 15.3-42 Gating timer 2 with Enable of timer 1

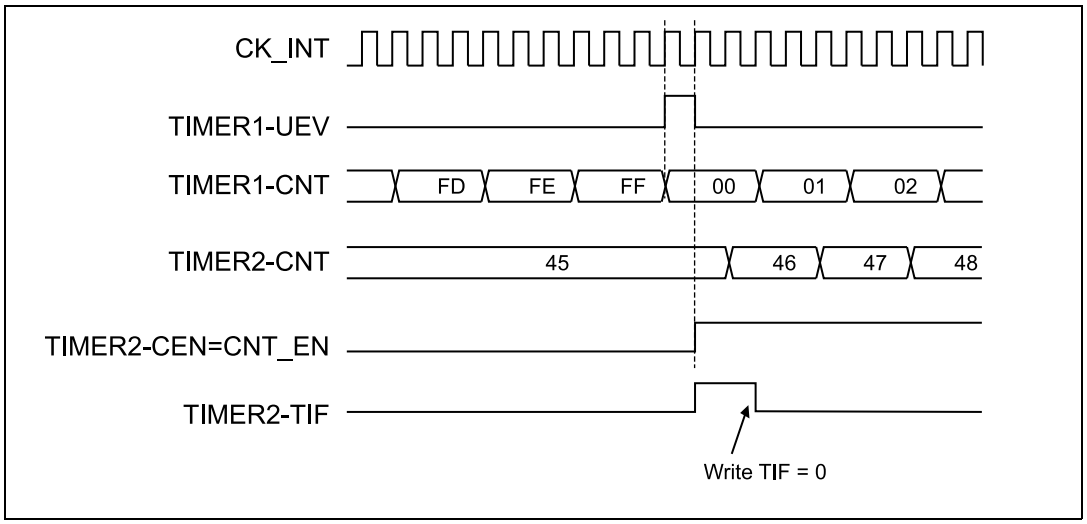


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to Figure 15.3-40 for connections. Timer 2 starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

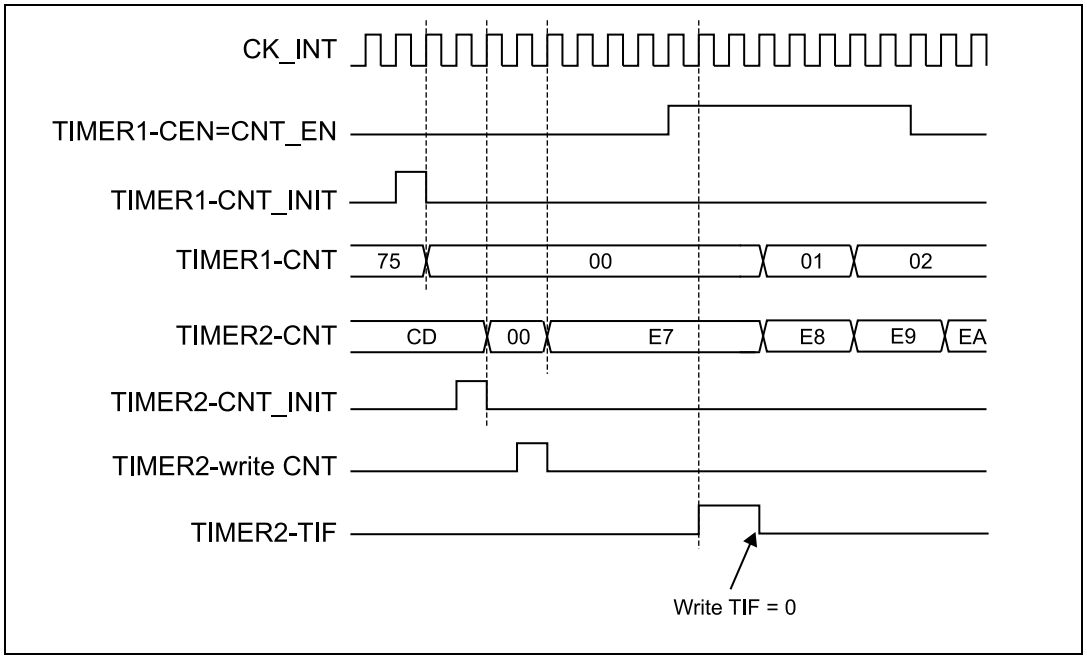
- Configure Timer 1 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM1_CR2 register).
- Configure the Timer 1 period (TIM1_ARR registers).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in trigger mode (SMS=110 in TIM2_SMCR register).
- Start Timer 1 by writing ‘1 in the CEN bit (TIM1_CR1 register)

Fig 15.3-43 Triggering timer 2 with update of timer 1



As in the previous example, the user can initialize both counters before starting counting. Figure 15.3-44 shows the behavior with the same configuration as in Figure 15.3-43 but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Fig 15.3-44 Triggering timer 2 with Enable of timer 1



Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of timer 1 when its TI1 input rises, and the enable of Timer 2 with the enable of Timer 1. Refer to Figure 15.3-40 for connections. To ensure the counters are aligned, Timer 1 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to Timer 2):

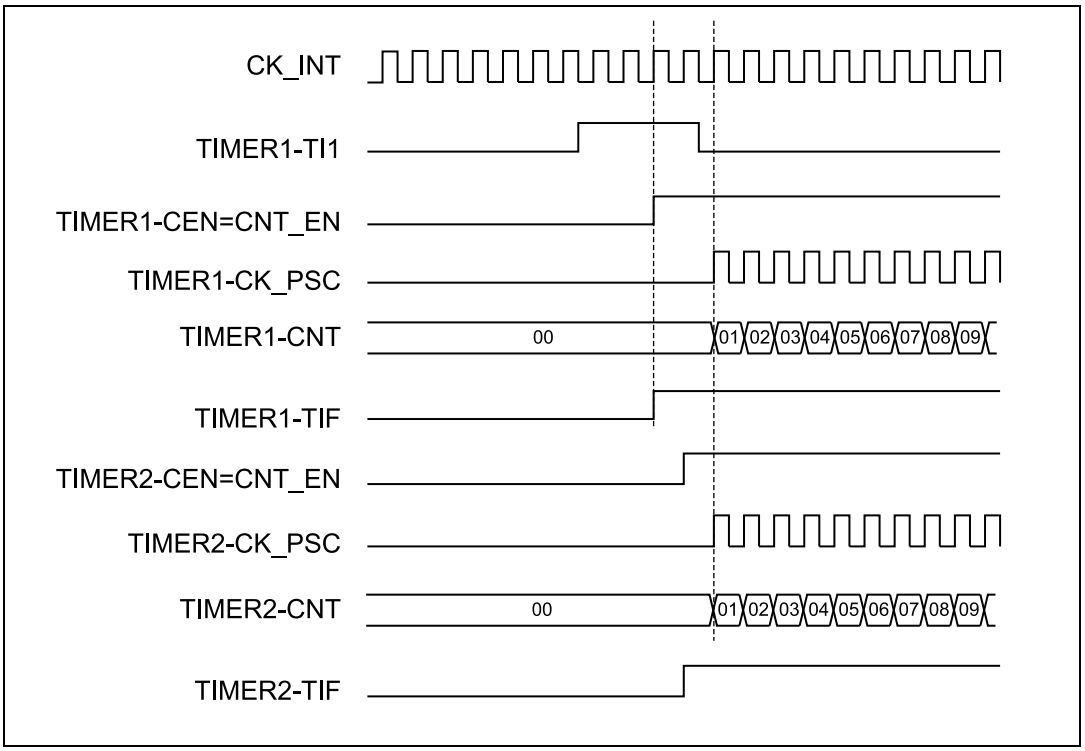
- Configure Timer 1 master mode to send its Enable as trigger output (MMS=001 in the TIM1_CR2 register).

- Configure Timer 1 slave mode to get the input trigger from TI1 (TS=100 in the TIM1_SMCR register).
- Configure Timer 1 in trigger mode (SMS=110 in the TIM1_SMCR register).
- Configure the Timer 1 in Master/Slave mode by writing MSM=1 (TIM1_SMCR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TS=000 in the TIM2_SMCR register).
- Configure Timer 2 in trigger mode (SMS=110 in the TIM2_SMCR register)

When a rising edge occurs on TI1 (Timer 1), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: *In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but you can easily insert an offset between them by writing any of the counter registers (TIMx_CNT). You can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on timer 1.*

Fig 15.3-45 Triggering timer 1 and 2 with timer 1 TI1 input



15.3.16 Debug mode

When the microcontroller enters debug mode (Cortex-M3 core-halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module.

15.4 TIMx Register

Tab 15.4-1 TIMx register map

偏移量	Register	Reset value	Description
TIM2_BA = 0x4000_0000 TIM3_BA = 0x4000_0400 TIM4_BA = 0x4000_0800 TIM5_BA = 0x4000_0C00			
0x00	TIMx_CR1	0x0000_0000	TIMx control register 1
0x04	TIMx_CR2	0x0000_0000	TIMx control register 2
0x08	TIMx_SMCR	0x0000_0000	TIMx slave mode control register
0x0C	TIMx_DIER	0x0000_0000	TIMx DMA/Interrupt enable register
0x10	TIMx_SR	0x0000_0000	TIMx status register
0x14	TIMx_EGR	0x0000_0000	TIMx event generation register
0x18	TIMx_CCMR1	0x0000_0000	TIMx capture/compare mode register 1
0x1C	TIMx_CCMR2	0x0000_0000	TIMx capture/compare mode register 2
0x20	TIMx_CCER	0x0000_0000	TIMx capture/compare enable register
0x24	TIMx_CNT	0x0000_0000	TIMx counter
0x28	TIMx_PSC	0x0000_0000	TIMx prescaler
0x2C	TIMx_ARR	0x0000_0000	TIMx auto-reload register
0x34	TIMx_CCR1	0x0000_0000	TIMx capture/compare register 1
0x38	TIMx_CCR2	0x0000_0000	TIMx capture/compare register 2
0x3C	TIMx_CCR3	0x0000_0000	TIMx capture/compare register 3
0x40	TIMx_CCR4	0x0000_0000	TIMx capture/compare register 4
0x48	TIMx_DCR	0x0000_0000	TIMx DMA control register
0x4C	TIMx_DMAR	0x0000_0000	TIMx DMA address for full transfer

15.4.1 TIMx control register 1 (TIMx_CR1)

Register	Address offset	Access	Reset value	Description
TIMx_CR1	0x00	RW	0x0000_0000	TIMx control register 1

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						CKD[1:0]	
7	6	5	4	3	2	1	0
ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN

TIMx control register 1 (TIMx_CR1)bit description

Bit	Access	Description
[31:10]	R	Reserved, must be kept at reset value.
[9:8]	RW	**CKD[1:0]:** Clock division This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx), 00: $t_{DTS} = t_{CK\_INT}$ 01: $t_{DTS} = 2 \times t_{CK\_INT}$ 10: $t_{DTS} = 4 \times t_{CK\_INT}$ 11: Reserved
[7]	RW	**ARPE:** Auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
[6:5]	RW	**CMS[1:0]:** Center-aligned mode selection 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR). 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down. 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up. 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down. **Note:** *It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)*
[4]	RW	**DIR:** Direction 0: Counter used as upcounter 1: Counter used as downcounter **Note:** *Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.*
[3]	RW	**OPM:** One-pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit CEN)

[2]	RW	**URS:** Update request source This bit is set and cleared by software to select the UEV event sources. 0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be: • Counter overflow/underflow • Setting the UG bit • Update generation through the slave mode controller 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.
[1]	RW	**UDIS:** Update disable This bit is set and cleared by software to enable/disable UEV event generation. 0: UEV enabled. The Update (UEV) event is generated by one of the following events: • Counter overflow/underflow • Setting the UG bit • Update generation through the slave mode controller Buffered registers are then loaded with their preload values. 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.
[0]	RW	**CEN:** Counter enable 0: Counter disabled 1: Counter enabled **Note:** *External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware* CEN is cleared automatically in one-pulse mode, when an update event occurs.

15.4.2 TIMx control register 2(TIMx_CR2)

Register	Address offset	Access	Reset value	Description
TIMx_CR2	0x04	RW	0x0000_0000	TIMx control register 2

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
TI1S	MMS[2:0]			CCDS	Reserved		

TIMx control register 2(TIMx_CR2)bit description

<table>
<tr><th>Bit</th><th>Access</th><th>Description</th></tr>
<tr><td>[31:8]</td><td>R</td><td>Reserved, must be kept at reset value.</td></tr>
<tr><td>[7]</td><td>RW</td><td>TI1S: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)</td></tr>
<tr><td>[6:4]</td><td>RW</td><td>MMS[2:0]: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
<table>
<tr><th>bit</th><th>mode</th><th>description</th></tr>
<tr><td>000:</td><td>Reset</td><td>the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.</td></tr>
<tr><td>001:</td><td>Enable</td><td>the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).</td></tr>
<tr><td>010:</td><td>Update</td><td>The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.</td></tr>
<tr><td>011:</td><td>Compare Pulse</td><td>The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)</td></tr>
<tr><td>100:</td><td>Compare</td><td>OC1REF signal is used as trigger output (TRGO)</td></tr>
<tr><td>101:</td><td>Compare</td><td>OC2REF signal is used as trigger output (TRGO)</td></tr>
<tr><td>110:</td><td>Compare</td><td>OC3REF signal is used as trigger output (TRGO)</td></tr>
<tr><td>111:</td><td>Compare</td><td>OC4REF signal is used as trigger output (TRGO)</td></tr>
</table>
Note : Note: The clock of the slave timer and ADC must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.</td></tr>
<tr><td>[3]</td><td>RW</td><td>CCDS: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs</td></tr>
<tr><td>[2:0]</td><td>R</td><td>Reserved, must be kept at reset value.</td></tr>
</table>

15.4.3 TIMx slave mode control register(TIMx_SMCR)

Register	Address offset	Access	Reset value	Description
TIMx_SMCR	0x08	RW	0x0000_0000	TIMx slave mode control register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ETP	ECE	ETPS[1:0]		ETF[3:0]			
7	6	5	4	3	2	1	0
MSM	TS[2:0]			Reserved	SMS[2:0]		

TIMx slave mode control register(TIMx_SMCR)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15]	RW	**ETP:** External trigger polarity This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations 0: ETR is non-inverted, active at high level or rising edge 1: ETR is inverted, active at low level or falling edge
[14]	RW	**ECE:** External clock enable This bit enables External clock mode 2. 0: External clock mode 2 disabled 1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal. **Note:** 1. *Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).* 2. *It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).* 3. *If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*
[13:12]	RW	**ETPS[1:0]:** External trigger prescaler External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks. 00: Prescaler OFF 01: ETRP frequency divided by 2 10: ETRP frequency divided by 4 11: ETRP frequency divided by 8

<table>
<tr><td>[11:8]</td><td>RW</td><td>ETF[3:0]: External trigger filter
This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS} 0001: $f_{SAMPLING} = f_{CK\_INT}$, N=2
0010: $f_{SAMPLING} = f_{CK\_INT}$, N = 4 0011: $f_{SAMPLING} = f_{CK\_INT}$, N = 8^2
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6 0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6^2 0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6 1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5 1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8 1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6 1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8</td></tr>
<tr><td>[7]</td><td>RW</td><td>MSM: Master/Slave mode
0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.</td></tr>
<tr><td>[6:4]</td><td>RW</td><td>TS[2:0]: Trigger selection
This bit-field selects the trigger input to be used to synchronize the counter.
000: Internal Trigger 0 (ITR0). 001: Internal Trigger 1 (ITR1).
010: Internal Trigger 2 (ITR2) 011: Internal Trigger 3 (ITR3)
100: TI1 Edge Detector (TI1F_ED) 101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2) 111: External Trigger input (ETRF)

<table>
<tr><th>Slave TIM</th><th>ITR0(TS = 000)</th><th>ITR1 (TS = 001)</th><th>ITR2(TS = 010)</th><th>ITR3(TS = 011)</th></tr>
<tr><td>TIM2</td><td>TIM1</td><td>TIM8</td><td>TIM3</td><td>TIM4</td></tr>
<tr><td>TIM3</td><td>TIM1</td><td>TIM2</td><td>TIM5</td><td>TIM4</td></tr>
<tr><td>TIM4</td><td>TIM1</td><td>TIM2</td><td>TIM3</td><td>TIM8</td></tr>
<tr><td>TIM5</td><td>TIM2</td><td>TIM2</td><td>TIM4</td><td>TIM8</td></tr>
</table>
Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.</td></tr>
<tr><td>[3]</td><td>R</td><td>Reserved, must be kept at reset value.</td></tr>
</table>

<table>
<tr><td>[2:0]</td><td>RW</td><td>SMS[2:0]: Slave mode selection
When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.)
<table>
<tr><th>SMS</th><th>mode</th><th>description</th></tr>
<tr><td>000:</td><td>Slave mode disabled</td><td>if CEN = ‘1 then the prescaler is clocked directly by the internal clock</td></tr>
<tr><td>001:</td><td>Encoder mode 1</td><td>Counter counts up/down on TI2FP1 edge depending on TI1FP2 level</td></tr>
<tr><td>010:</td><td>Encoder mode 2</td><td>Counter counts up/down on TI1FP2 edge depending on TI2FP1 level.</td></tr>
<tr><td>011:</td><td>Encoder mode 3</td><td>Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input</td></tr>
<tr><td>100:</td><td>Reset Mode</td><td>Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.</td></tr>
<tr><td>101:</td><td>Gated Mode</td><td>The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled</td></tr>
<tr><td>110:</td><td>Trigger Mode</td><td>The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.</td></tr>
<tr><td>111:</td><td>External Clock Mode 1</td><td>Rising edges of the selected trigger (TRGI) clock the counter</td></tr>
</table>
Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal. The clock of the slave timer must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.</td></tr>
</table>

15.4.4 TIMx DMA/Interrupt enable register(TIMx_DIER)

Register	Address offset	Access	Reset value	Description
TIMx_DIER	0x0C	RW	0x0000_0000	TIMx DMA/Interrupt enable register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	TDE	Reserved	CC4DE	CC3DE	CC2DE	CC1DE	UDE
7	6	5	4	3	2	1	0
Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE

TIMx DMA/Interrupt enable register(TIMx_DIER)bit description

Bit	Access	Description
[31:15]	R	Reserved, must be kept at reset value.

[14]	RW	**TDE:** Trigger DMA request enable 0: Trigger DMA request disabled. 1: Trigger DMA request enabled.
[13]	R	Reserved, must be kept at reset value.
[12]	RW	**CC4DE:** Capture/Compare 4 DMA request enable 0: CC4 DMA request disabled. 1: CC4 DMA request enabled.
[11]	RW	**CC3DE:** Capture/Compare 3 DMA request enable 0: CC3 DMA request disabled. 1: CC3 DMA request enabled
[10]	RW	**CC2DE:** Capture/Compare 2 DMA request enable 0: CC2 DMA request disabled. 1: CC2 DMA request enabled.
[9]	RW	**CC1DE:** Capture/Compare 1 DMA request enable 0: CC1 DMA request disabled. 1: CC1 DMA request enabled
[8]	RW	**UDE:** Update DMA request enable 0: Update DMA request disabled. 1: Update DMA request enabled
[7]	R	Reserved, must be kept at reset value.
[6]	RW	**TIE:** Trigger interrupt enable 0: Trigger interrupt disabled. 1: Trigger interrupt enabled.
[5]	R	Reserved, must be kept at reset value.
[4]	RW	**CC4IE:** Capture/Compare 4 interrupt enable 0: CC4 interrupt disabled. 1: CC4 interrupt enabled.
[3]	RW	**CC3IE:** Capture/Compare 3 interrupt enable 0: CC3 interrupt disabled. 1: CC3 interrupt enabled.
[2]	RW	**CC2IE:** Capture/Compare 2 interrupt enable 0: CC2 interrupt disabled. 1: CC2 interrupt enabled.
[1]	RW	**CC1IE:** Capture/Compare 1 interrupt enable 0: CC1 interrupt disabled. 1: CC1 interrupt enabled.
[0]	RW	**UIE:** Update interrupt enable 0: Update interrupt disabled. 1: Update interrupt enabled.

15.4.5 TIMx status register(TIMx_SR)

Register	Address offset	Access	Reset value	Description
TIMx_SR	0x10	RC_W0	0x0000_0000	TIMx status register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved
7	6	5	4	3	2	1	0
Reserved	TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF

TIMx status register(TIMx_SR)bit description

Bit	Access	Description
[31:13]	R	Reserved, must be kept at reset value.
[12]	RC_W0	**CC4OF:** Capture/Compare 4 overcapture flag refer to CC1OF description
[11]	RC_W0	**CC3OF:** Capture/Compare 3 overcapture flag refer to CC1OF description
[10]	RC_W0	**CC2OF:** Capture/compare 2 overcapture flag refer to CC1OF description
[9]	RC_W0	**CC1OF:** Capture/Compare 1 overcapture flag This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’. 0: No overcapture has been detected. 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set
[8:7]	R	Reserved, must be kept at reset value
[6]	RC_W0	**TIF:** Trigger interrupt flag This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software. 0: No trigger event occurred. 1: Trigger interrupt pending.
[5]	R	Reserved, must be kept at reset value
[4]	RC_W0	**CC4IF:** Capture/Compare 4 interrupt flag refer to CC1IF description
[3]	RC_W0	**CC3IF:** Capture/Compare 3 interrupt flag refer to CC1IF description
[2]	RC_W0	**CC2IF:** Capture/Compare 2 interrupt flag refer to CC1IF description

[1]	RC_W0	**CC1IF:** Capture/compare 1 interrupt flag **If channel CC1 is configured as output:** This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software. 0: No match. 1: The content of the counter TIMx_CNT has matched the content of the TIMx_CCR1 register. **If channel CC1 is configured as input:** This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register. 0: No input capture occurred. 1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity).
[0]	RC_W0	**UIF:** Update interrupt flag • This bit is set by hardware on an update event. It is cleared by software. 0: No update occurred. 1: Update interrupt pending. This bit is set by hardware when the registers are updated: • At overflow or underflow and if the UDIS=0 in the TIMx_CR1 register. • When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register. • When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register

15.4.6 TIMx event generation register (TIMx_EGR)

Register	Address offset	Access	Reset value	Description
TIMx_EGR	0x14	W	0x0000_0000	TIMx event generation register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG

TIMx event generation register (TIMx_EGR)bit description

Bit	Access	Description
[31:7]	R	Reserved, must be kept at reset value
[6]	W	**TG:** Trigger generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.
[5]	R	Reserved, must be kept at reset value.
[4]	W	**CC4G:** Capture/compare 4 generation refer to CC1G description
[3]	W	**CC3G:** Capture/compare 3 generation refer to CC1G description
[2]	W	**CC2G:** Capture/compare 2 generation refer to CC1G description
[1]	W	**CC1G:** Capture/compare 1 generation This bit is set by software in order to generate an event, it is automatically cleared by hardware. 0: No action 1: A capture/compare event is generated on channel 1: **If channel CC1 is configured as output:** CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled. **If channel CC1 is configured as input:** The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.
[0]	W	**UG:** Update generation This bit can be set by software, it is automatically cleared by hardware. 0: No action 1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

15.4.7 TIMx capture/compare mode register 1(TIMx_CCMR1)

Note: *The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. Take care that the same bit can have a different meaning for the input stage and for the output stage.*

Register	Address offset	Access	Reset value	Description
TIMx_CCMR1	0x18	RW	0x0000_0000	TIMx capture/compare mode register 1

31	30	29	28	27	26	25	24
Reserved							
Reserved							
23	22	21	20	19	18	17	16
Reserved							
Reserved							
15	14	13	12	11	10	9	8
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]	
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]	
7	6	5	4	3	2	1	0
IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	

TIMx_CCMR1 Input capture mode:

TIMx capture/compare mode register 1(TIMx_CCMR1)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value
[15:12]	RW	**IC2F[3:0]:** Input capture 2 filter
[11:10]	RW	**IC2PSC[1:0]:** Input capture 2 prescaler
[9:8]	RW	**CC2S[1:0]:** Capture/compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER)*

[7:4]	RW	**IC1F[3:0]:** Input capture 1 filter This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output: 0000: No filter, sampling is done at f_{DTS} 0001: $f_{SAMPLING} = f_{CK\_INT}$, N=2 0010: $f_{SAMPLING} = f_{CK\_INT}$, N = 4 0011: $f_{SAMPLING} = f_{CK\_INT}$, N = 8 0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6 0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8 0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6 0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8 1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6 1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8 1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5 1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6 1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8 1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5 1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6 1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8
[3:2]	RW	**IC1PSC[1:0]:** Input capture 1 prescaler This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register). 00: no prescaler, capture is done each time an edge is detected on the capture input 01: capture is done once every 2 events 10: capture is done once every 4 events 11: capture is done once every 8 events
[1:0]	RW	**CC1S[1:0]:** Capture/Compare 1 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).*

TIMx_CCMR1 Output compare mode:

TIMx capture/compare mode register 1(TIMx_CCMR1)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value
[15]	RW	**OC2CE:** Output compare 2 clear enable
[14:12]	RW	**OC2M[2:0]:** Output compare 2 mode
[11]	RW	**OC2PE:** Output compare 2 preload enable
[10]	RW	**OC2FE:** Output compare 2 fast enable
[9:8]	RW	**CC2S[1:0]:** Capture/Compare 2 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register) **Note:** *CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).*

[7]	RW	**OC1CE:** Output compare 1 clear enable OC1CE: Output Compare 1 Clear Enable 0: OC1Ref is not affected by the ETRF input 1: OC1Ref is cleared as soon as a High level is detected on ETRF input
[6:4]	RW	**OC1M[2:0]:** Output compare 1 mode These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits. 000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base). 001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1). 010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1). 011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1. 100: Force inactive level - OC1REF is forced low. 101: Force active level - OC1REF is forced high. 110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF= ‘0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1). 111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive **Note:** *In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode*
[3]	RW	**OC1PE:** Output compare 1 preload enable 0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately. 1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event. **Note:** *1: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).* *2: The PWM mode can be used without validating the preload register only in one- pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed*
[2]	RW	**OC1FE:** Output compare 1 fast enable This bit is used to accelerate the effect of an event on the trigger in input on the CC output. 0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles. 1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

[1:0]	RW	**CC1S[1:0]:** Capture/Compare 1 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).*

15.4.8 TIMx capture/compare mode register 2(TIMx_CCMR2)

Note: *Refer to the above CCMR1 register description.*

Register	Address offset	Access	Reset value	Description
TIMx_CCMR2	0x1C	RW	0x0000_0000	TIMx capture/compare mode register 2

31	30	29	28	27	26	25	24
Reserved							
Reserved							
23	22	21	20	19	18	17	16
Reserved							
Reserved							
15	14	13	12	11	10	9	8
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]	
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]	
7	6	5	4	3	2	1	0
IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	

TIMx_CCMR2 Input capture mode:

TIMx capture/compare mode register 2(TIMx_CCMR2)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value
[15:12]	RW	**IC4F[3:0]:** Input capture 4 filter
[11:10]	RW	**IC4PSC[1:0]:** Input capture 4 prescaler
[9:8]	RW	**CC4S[1:0]:** Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).*
[7:4]	RW	**IC3F[3:0]:** Input capture 3 filter
[3:2]	RW	**IC3PSC[1:0]:** Input capture 3 prescaler

[1:0]	RW	**CC3S[1:0]:** Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).*

TIMx_CCMR2 Output compare mode:

TIMx capture/compare mode register 2(TIMx_CCMR2)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value
[15]	RW	**OC4CE:** Output compare 4 clear enable
[14:12]	RW	**OC4M[2:0]:** Output compare 4 mode
[11]	RW	**OC4PE:** Output compare 4 preload enable
[10]	RW	**OC4FE:** Output compare 4 fast enable
[9:8]	RW	**CC4S[1:0]:** Capture/Compare 4 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER)*
[7]	RW	**OC3CE:** Output compare 3 clear enable
[6:4]	RW	**OC3M[2:0]:** Output compare 3 mode
[3]	RW	**OC3PE:** Output compare 3 preload enable
[2]	RW	**OC3FE:** Output compare 3 fast enable
[1:0]	RW	**CC3S[1:0]:** Capture/Compare 3 selection This bit-field defines the direction of the channel (input/output) as well as the used input. 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 is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register) **Note:** *CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).*

15.4.9 TIMx capture/compare enable register(TIMx_CCER)

Register	Address offset	Access	Reset value	Description
TIMx_CCER	0x20	RW	0x0000_0000	TIMx capture/compare enable register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		CC4P	CC4E	Reserved		CC3P	CC3E
7	6	5	4	3	2	1	0
Reserved		CC2P	CC2E	Reserved		CC1P	CC1E

TIMx capture/compare enable register(TIMx_CCER)bit description

Bit	Access	Description
[31:14]	R	Reserved, must be kept at reset value
[13]	RW	**CC4P:** Capture/Compare 4 output polarity refer to CC1P description
[12]	RW	**CC4E:** Capture/Compare 4 output enable refer to CC1E description
[11:10]	R	Reserved, must be kept at reset value
[9]	RW	**CC3P:** Capture/Compare 3 output polarity refer to CC1P description
[8]	RW	**CC3E:** Capture/Compare 3 output enable refer to CC1E description
[7:6]	R	Reserved, must be kept at reset value
[5]	RW	**CC2P:** Capture/Compare 2 output polarity refer to CC1P description
[4]	RW	**CC2E:** Capture/Compare 2 output enable refer to CC1E description
[3:2]	R	Reserved, must be kept at reset value.
[1]	RW	**CC1P:** Capture/Compare 1 output polarity **CC1 channel configured as output:** 0: OC1 active high. 1: OC1 active low. **CC1 channel configured as input:** This bit selects whether IC1 or IC1 is used for trigger or capture operations. 0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted. 1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted.

[0]	RW	**CC1E:** Capture/Compare 1 output enable **CC1 channel configured as output:** 0: Off - OC1 is not active. 1: On - OC1 signal is output on the corresponding output pin. **CC1 channel configured as input:** This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not. 0: Capture disabled. 1: Capture enabled.

Tab 15.4-13 Output control bit for standard OCx channels

CCxE	bit OCx output state
0	Output Disabled (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Polarity, OCx_EN=1

Note: *The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO and AFIO registers*

15.4.10 TIMx counter (TIMx_CNT)

Register	Address offset	Access	Reset value	Description
TIMx_CNT	0x24	RW	0x0000_0000	TIMx counter

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

TIMx counter (TIMx_CNT)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**CNT[19:0]:** Counter value

15.4.11 TIMx prescale(TIMx_PSC)

Register	Address offset	Access	Reset value	Description
TIMx_PSC	0x28	RW	0x0000_0000	TIMx prescale

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

TIMx prescale(TIMx_PSC)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value
[15:0]	RW	**PSC[15:0]:** Prescaler value The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1). PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

15.4.12 TIMx auto-reload register(TIMx_ARR)

Register	Address offset	Access	Reset value	Description
TIMx_ARR	0x2C	RW	0x0000_0000	TIMx auto-reload register

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

TIMx auto-reload register (TIMx_ARR)bit description

Bit	Access	Description
[31:0]	RW	**ARR[31:0]:** ARR is the value to be loaded in the actual auto-reload register. Refer to the Section 15.3.1: Time-base unit for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.

15.4.13 TIMx capture/compare register 1(TIMx_CCR1)

Register	Address offset	Access	Reset value	Description
TIMx_CCR1	0x34	RW	0x0000_0000	TIMx capture/compare register 1

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CCR1[15:8]							
7	6	5	4	3	2	1	0
CCR1[7:0]							

TIMx capture/compare register 1(TIMx_CCR1)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**CCR1[15:0]:** Capture/Compare 1 value **If channel CC1 is configured as output:** CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output. **If channel CC1is configured as input:** CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

15.4.14 TIMx capture/compare register 2(TIMx_CCR2)

Register	Address offset	Access	Reset value	Description
TIMx_CCR2	0x38	RW	0x0000_0000	TIMx capture/compare register 2

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CCR2[15:8]							
7	6	5	4	3	2	1	0
CCR2[7:0]							

TIMx capture/compare register 2(TIMx_CCR2)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**CCR2[15:0]:** Capture/Compare 2 value **If channel CC2 is configured as output:** CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output. **If channel CC2 is configured as input:** CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed

15.4.15 TIMx capture/compare register 3(TIMx_CCR3)

Register	Address offset	Access	Reset value	Description
TIMx_CCR3	0x3C	RW	0x0000_0000	TIMx capture/compare register 3

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CCR3[15:8]							
7	6	5	4	3	2	1	0
CCR3[7:0]							

TIMx capture/compare register 3(TIMx_CCR3)bit description

Bit	Access	Description
[31:16]	R	Reserved

[15:0]	RW	**CCR3[15:0]:** Capture/Compare value **If channel CC3 is configured as output:** CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output. **If channel CC3 is configured as input:** CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

15.4.16 TIMx capture/compare register 4(TIMx_CCR4)

Register	Address offset	Access	Reset value	Description
TIMx_CCR4	0x40	RW	0x0000_0000	TIMx capture/compare register 4

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CCR4[15:8]							
7	6	5	4	3	2	1	0
CCR4[7:0]							

TIMx capture/compare register 4(TIMx_CCR4)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**CCR4[15:0]:** Capture/Compare value **if CC4 channel is configured as output (CC4S bits):** CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output. **if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):** CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

15.4.17 TIMx DMA control register(TIMx_DCR)

Register	Address offset	Access	Reset value	Description
TIMx_DCR	0x48	RW	0x0000_0000	TIMx DMA control register

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

TIMx DMA control register(TIMx_DCR)bit description

Bit	Access	Description
[31:13]	R	Reserved,must be kept at reset value.
[12:8]	RW	**DBL[4:0]:** DMA burst length This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address). 00000 ~ 10001 : 1 transfer ~ 18 transfers
[7:5]	R	Reserved
[4: 0]	RW	**DBA[4:0]:** DMA base address This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register. Example: 00000: TIMx_CR1, 00001: TIMx_CR2, 00010: TIMx_SMCR ... **Example** *Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.*

15.4.18 TIMx DMA address for full transfer(TIMx_DMAR)

Register	Address offset	Access	Reset value	Description
TIMx_DMAR	0x4C	RW	0x0000_0000	TIMx DMA address for full transfer

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

TIMx DMA address for full transfer(TIMx_DMAR)bit description

Bit	Access	Description
[31:16]	R	Reserved

[15:0]	RW	**DDMABBL[15:0]:** DMA register for burst accesses A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4 where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

- Configure the corresponding DMA channel as follows:
  - DMA channel peripheral address is the DMAR register address
  - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
  - Number of data to transfer = 3 (See note below).
  - Circular mode disabled.
- Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
- Enable the TIMx update DMA request (set the UDE bit in the DIER register).
- Enable TIMx
- Enable the DMA channel

Note: *This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.*

16 Basic timers (TIM6 and TIM7)

16.1 TIM6 and TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 32-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

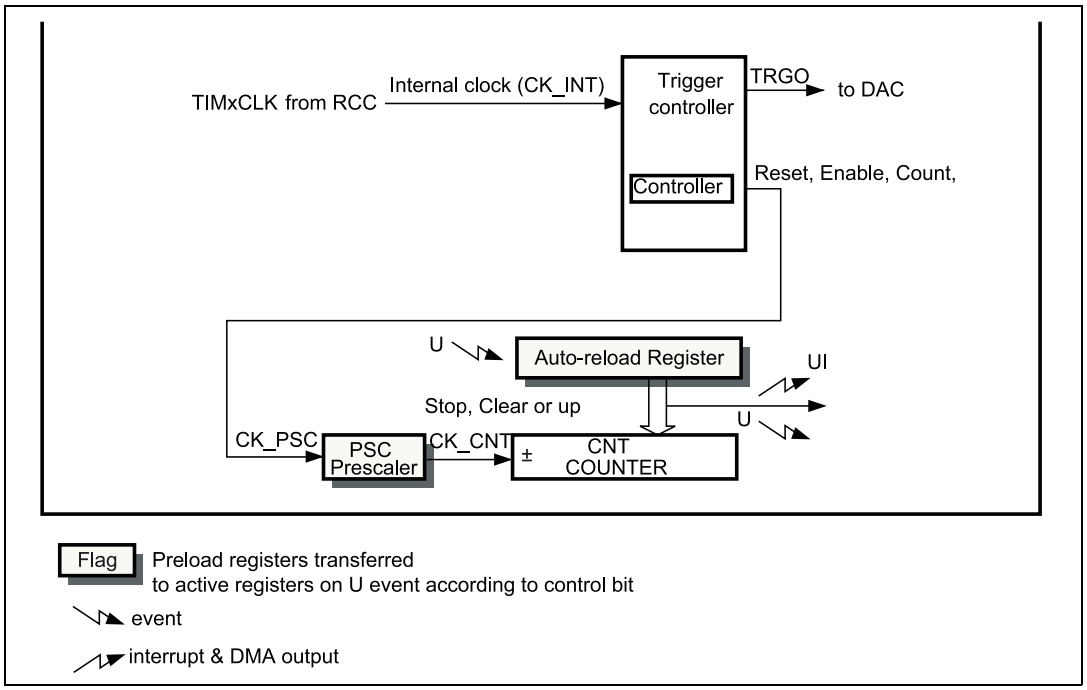
The timers are completely independent, and do not share any resources.

16.2 TIM6 and TIM7 main features

Basic timer (TIM6 and TIM7) features include:

- 32-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65536
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Fig 16.2-1 Basic timer block diagram



16.3 TIM6 and TIM7 functional description

16.3.1 Time-base unit

The main block of the programmable timer is a 32-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler. The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 16.3-1 and Figure 16.3-2 give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Fig 16.3-1 Counter timing diagram with prescaler division change from 1 to 2

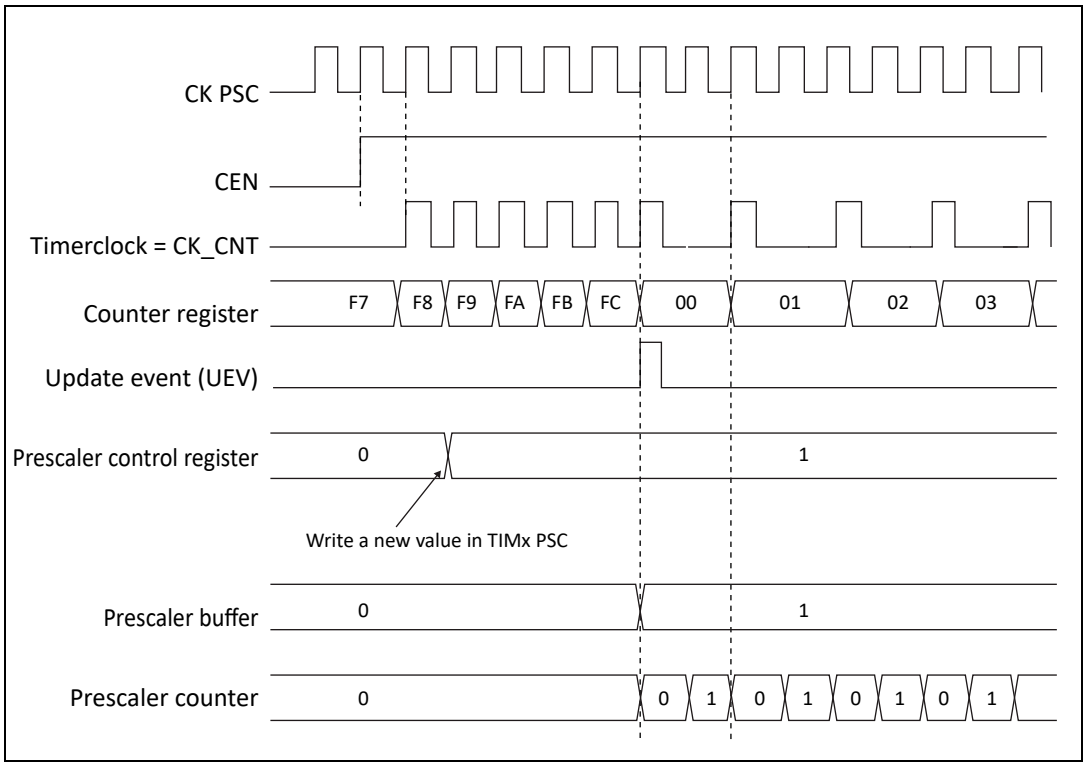
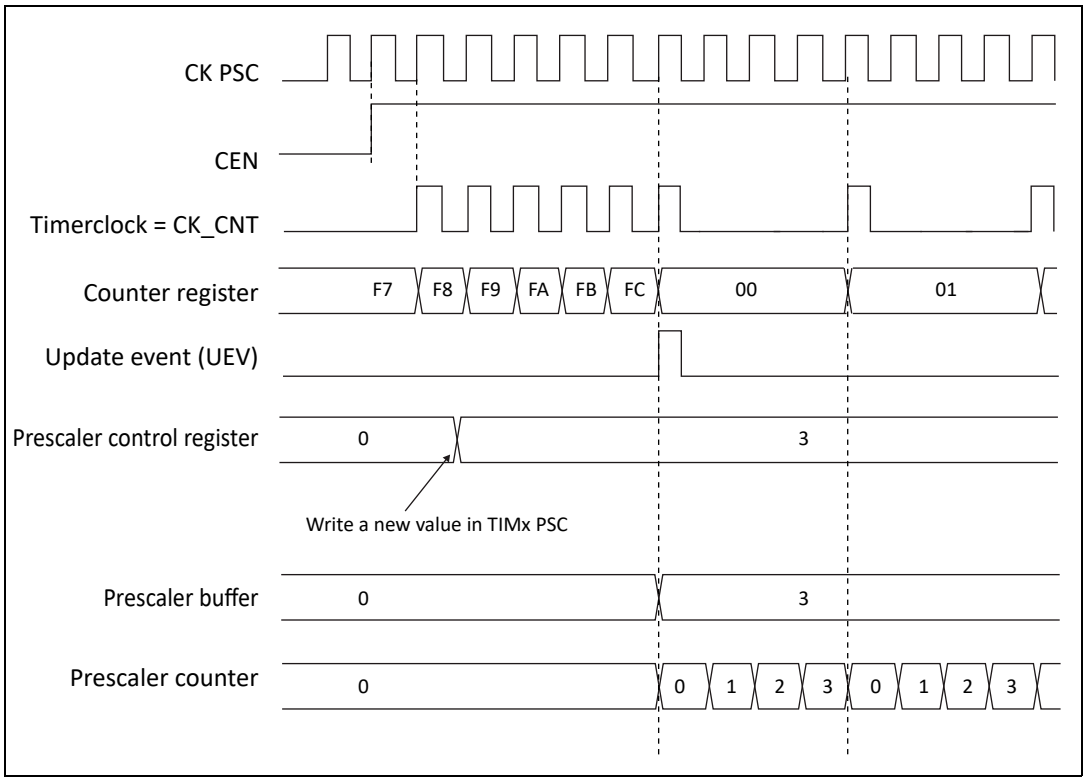


Fig 16.3-2 Counter timing diagram with prescaler division change from 1 to 4



16.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

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

Fig 16.3-3 Counter timing diagram, internal clock divided by 1

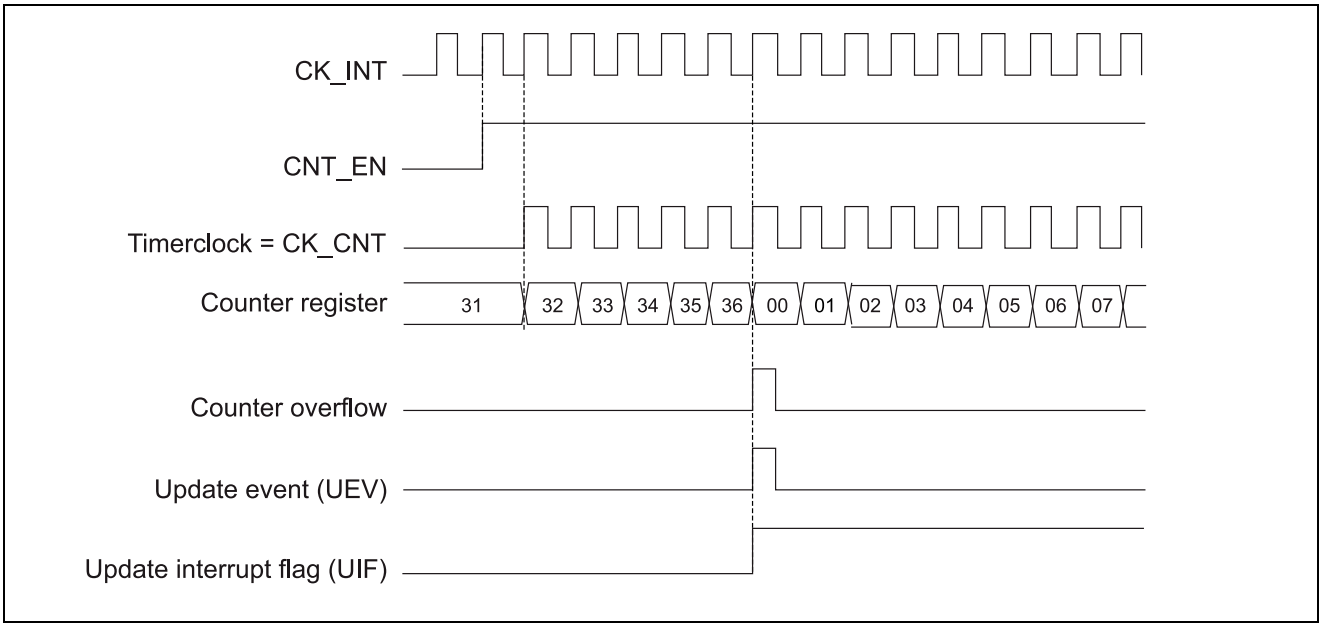


Fig 16.3-4 Counter timing diagram, internal clock divided by 2

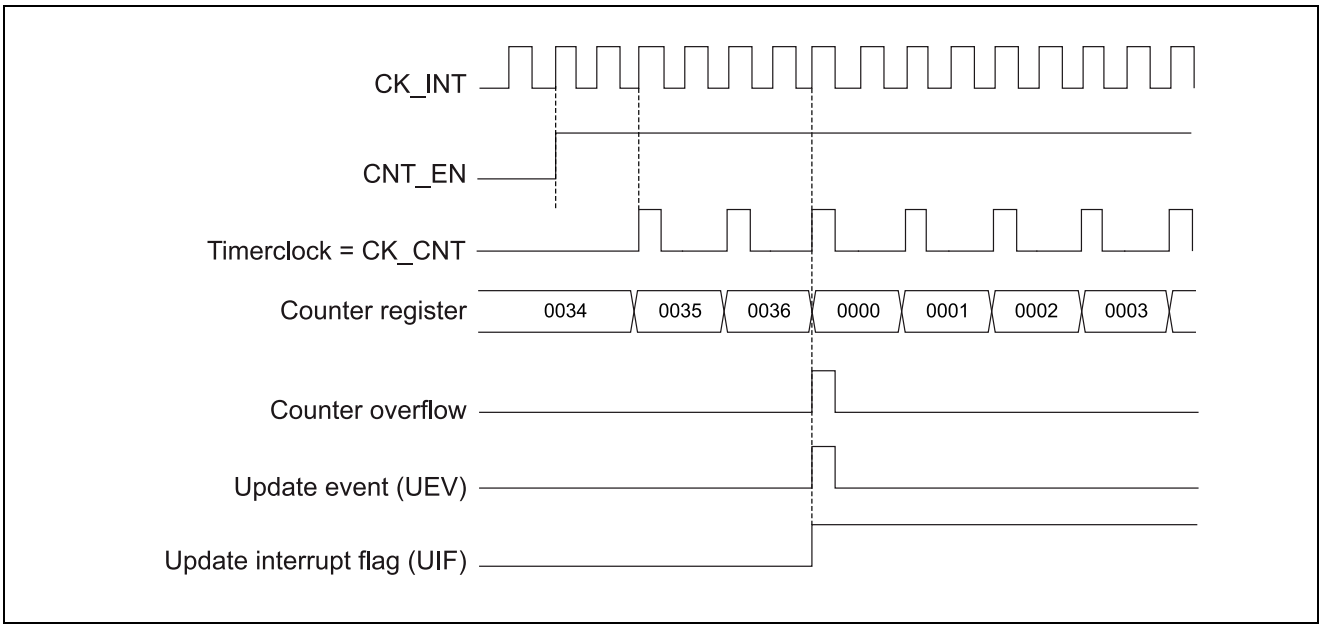


Fig 16.3-5 Counter timing diagram, internal clock divided by 4

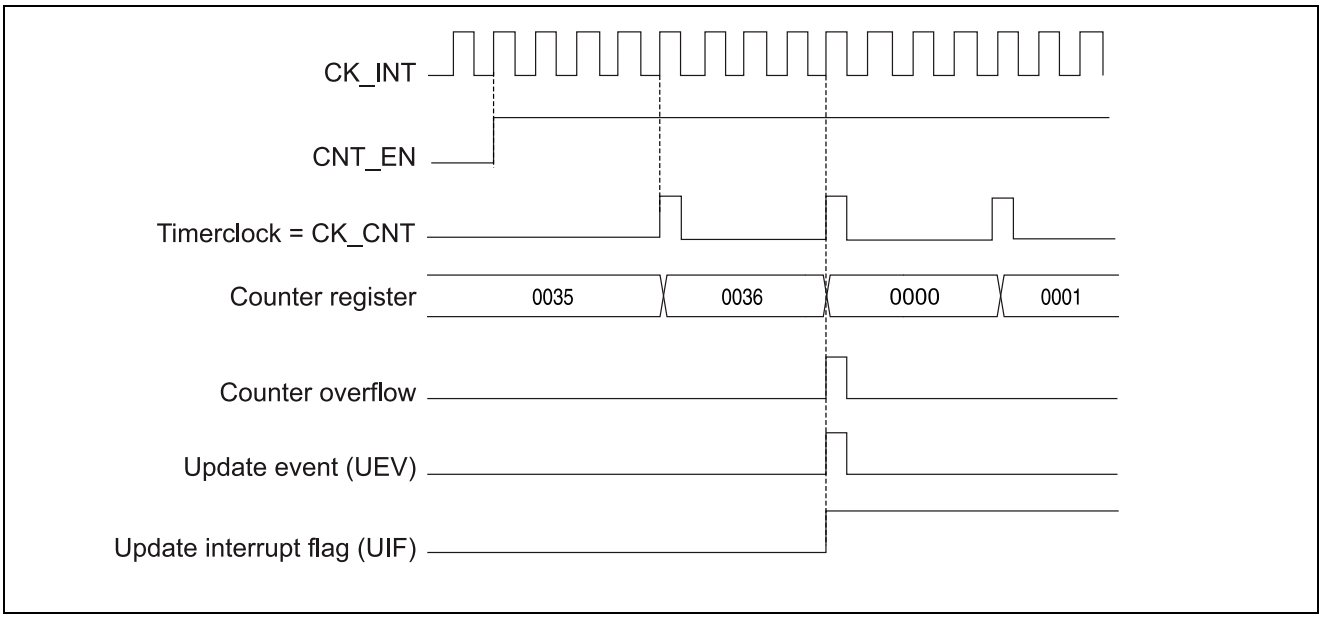


Fig 16.3-6 Counter timing diagram, internal clock divided by N

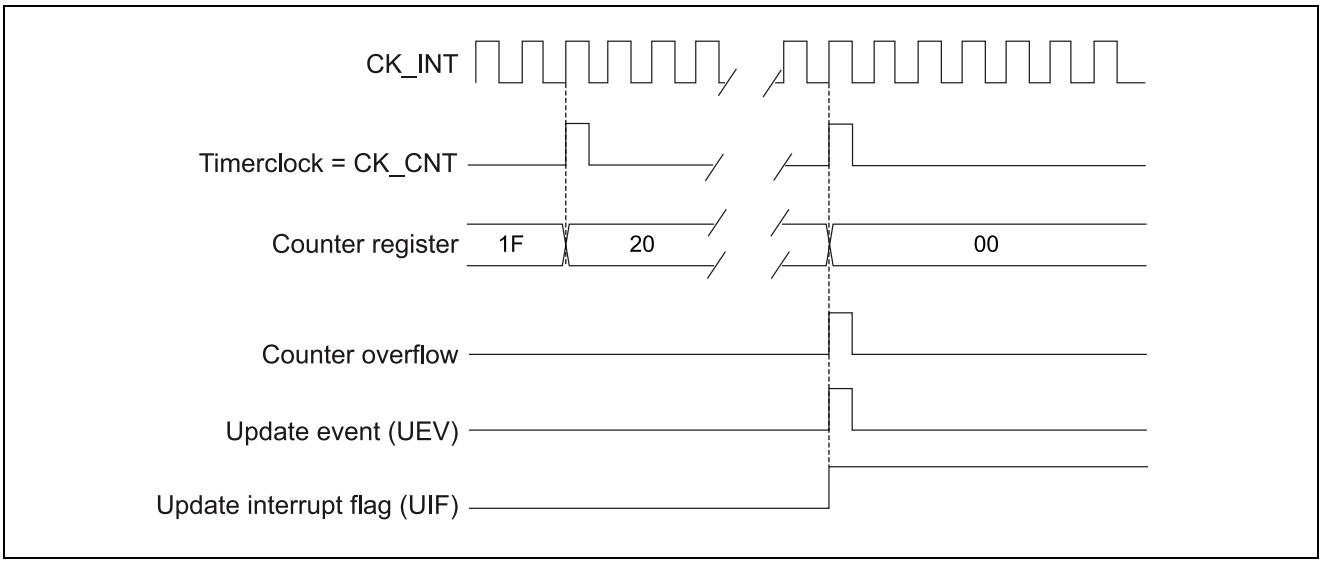


Fig 16.3-7 Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

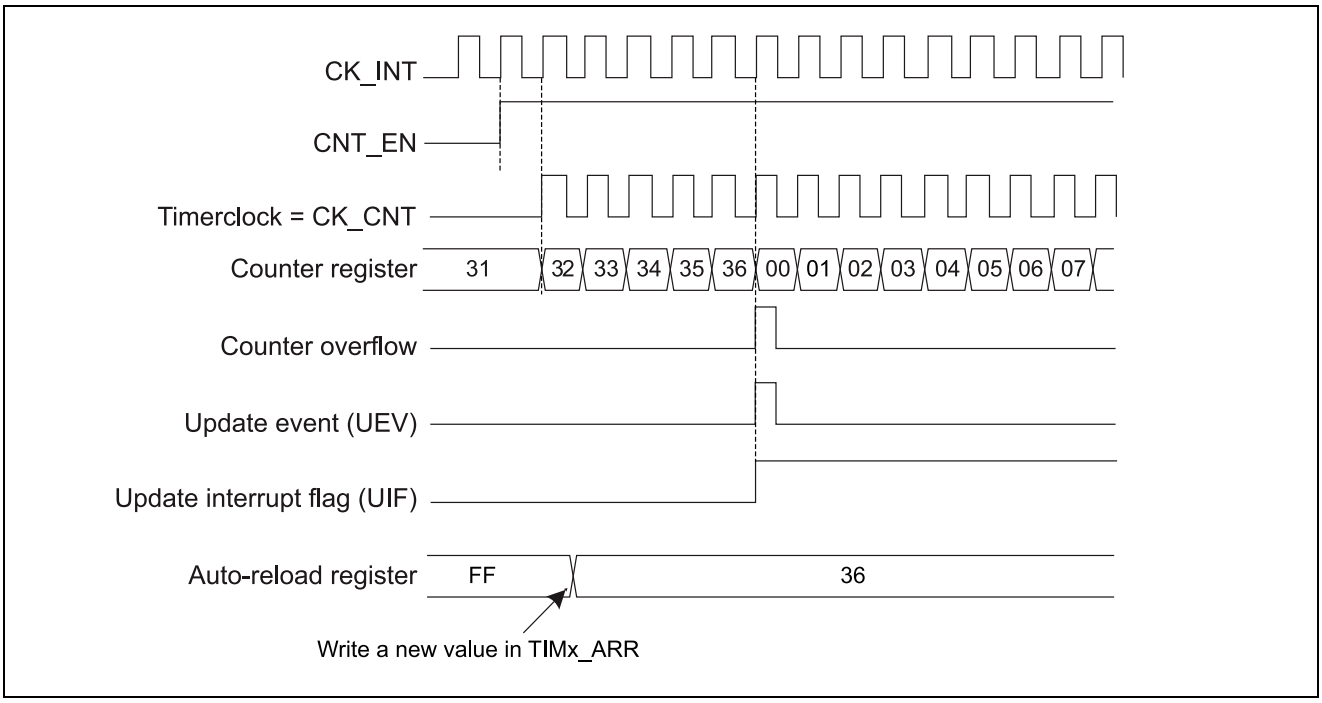
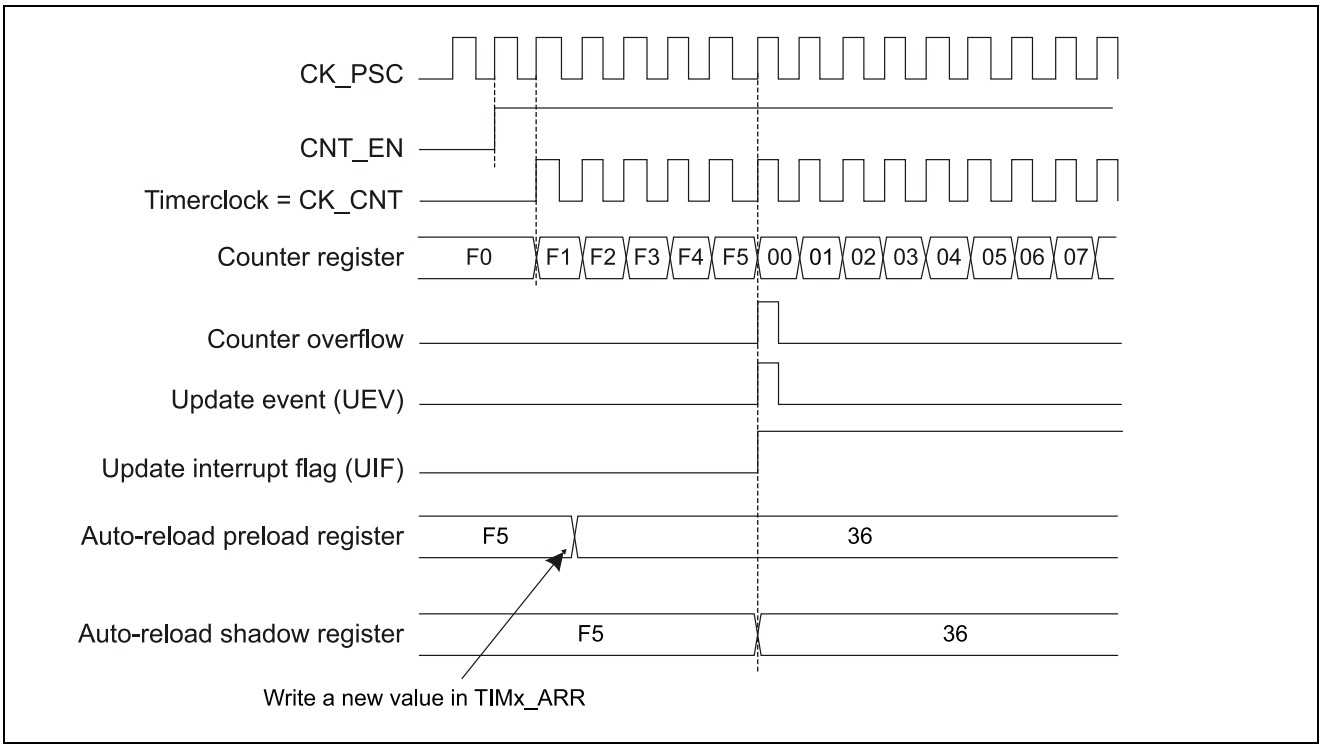


Fig 16.3-8 Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



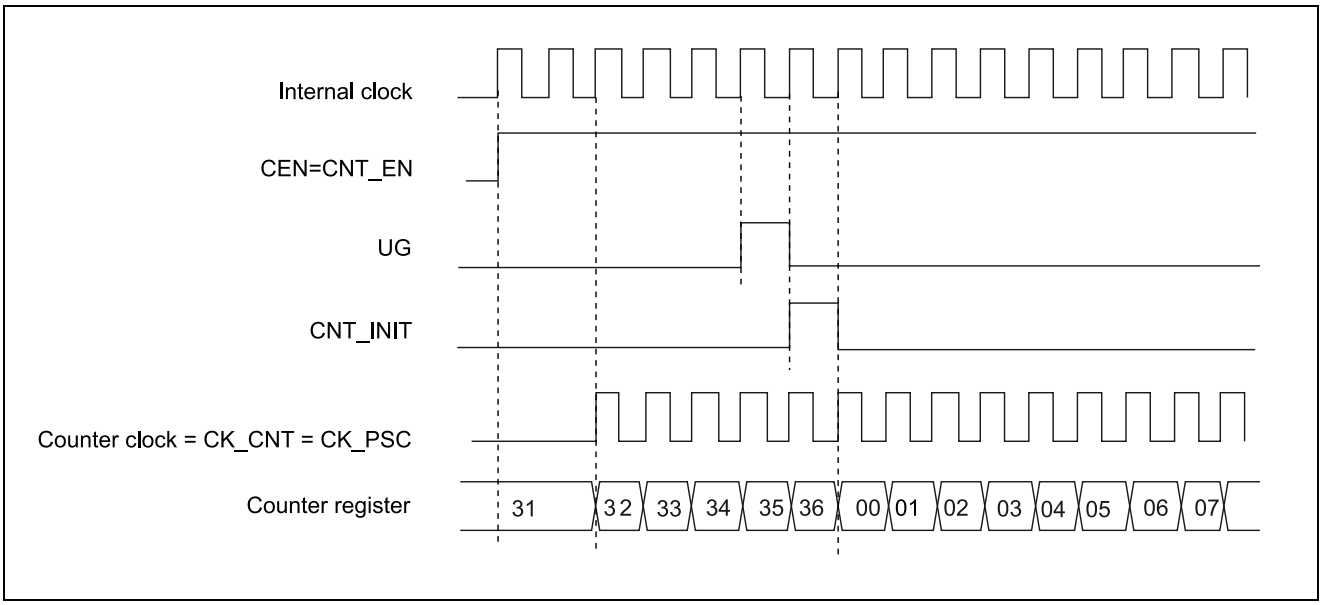
16.3.3 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 16.3-9 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Fig 16.3-9 Control circuit in normal mode, internal clock divided by 1



16.3.4 Debug mode

When the microcontroller enters the debug mode (Cortex-M3 core-halted), the TIMx counter either continues to work normally or stops, depending on the DBG_TIMx_STOP configuration bit in the DBG module.

16.3.5 TIM6 and TIM7 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits)

Tab 16.3-1 TIMx Register map

offset	Register	Reset value	Description
TIM6_BA = 0x4000_1000 TIM7_BA = 0x4000_1400			
0x00	TIMx_CR1	0x0000_0000	TIM6 and TIM7 control register 1
0x04	TIMx_CR2	0x0000_0000	TIM6 and TIM7 control register 2
0x0C	TIMx_DIER	0x0000_0000	TIM6 and TIM7 DMA/Interrupt enable register
0x10	TIMx_SR	0x0000_0000	TIM6 and TIM7 status register
0x14	TIMx_EGR	0x0000_0000	TIM6 and TIM7 event generation register
0x24	TIMx_CNT	0x0000_0000	TIM6 and TIM7 counter
0x28	TIMx_PSC	0x0000_0000	TIM6 and TIM7 prescaler
0x2C	TIMx_ARR	0x0000_0000	TIM6 and TIM7 auto-reload register

16.3.6 TIM6 and TIM7 control register 1(TIMx_CR1, x = 6,7)

Register	Address offset	Access	Reset value	Description
TIMx_CR1	0x00	RW	0x0000_0000	TIM6 and TIM7 control register 1

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

TIM6 and TIM7 control register 1(TIMx_CR1)Bit Description

Bit	Access	Description
[31:8]	R	Reserved, must be kept at reset value.
[7]	RW	**ARPE:** Auto-reload preload enable 0: TIMx_ARR register is not buffered. 1: TIMx_ARR register is buffered
[6:4]	R	Reserved, must be kept at reset value.
[3]	RW	**OPM:** One-pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the CEN bit).
[2]	RW	**URS:** Update request source This bit is set and cleared by software to select the UEV event sources. 0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be: • Counter overflow/underflow • Setting the UG bit • Update generation through the slave mode controller 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled
[1]	RW	**UDIS:** Update disable This bit is set and cleared by software to enable/disable UEV event generation. 0: UEV enabled. The Update (UEV) event is generated by one of the following events: • Counter overflow/underflow • Setting the UG bit • Update generation through the slave mode controller Buffered registers are then loaded with their preload values. 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

[0]	RW	**CEN:** Counter enable 0: Counter disabled 1: Counter enabled **Note:** *Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.* *CEN is cleared automatically in one-pulse mode, when an update event occurs.*

16.3.7 TIM6 and TIM7 control register 2 (TIMx_CR2, x = 6,7)

Register	Address offset	Access	Reset value	Description
TIMx_CR2	0x04	RW	0x0000_0000	TIM6 and TIM7 control register 2

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

TIM6 and TIM7 control register 2 (TIMx_CR2)Bit Description

Bit	Access	Description
[31:7]	R	Reserved, must be kept at reset value.
[6:4]	RW	**MMS[2:0]:** Master mode selection These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows: **MMS** / **Mode** / **Description** 000: Reset — the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset. 001: Enable — the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register). 010: Update — The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer. **Note:** *The clock of the slave timer and ADC must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*
[3:0]	R	Reserved, must be kept at reset value.

16.3.8 TIM6 and TIM7 DMA/Interrupt enable register (TIMx_DIER, x = 6,7)

Register	Address offset	Access	Reset value	Description
TIMx_DIER	0x0C	RW	0x0000_0000	TIM6 and TIM7 DMA/Interrupt enable register

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

TIM6 and TIM7 DMA/Interrupt enable register (TIMx_DIER)Bit Description

Bit	Access	Description
[31:9]	R	Reserved, must be kept at reset value。
[8]	RW	**UDE:** Update DMA request enable 0: Update DMA request disabled. 1: Update DMA request enabled
[7:1]	R	Reserved, must be kept at reset value
[0]	RW	**UIE:** Update interrupt enable 0: Update interrupt disabled. 1: Update interrupt enabled

16.3.9 TIM6 and TIM7 status register(TIMx_SR, x = 6,7)

Register	Address offset	Access	Reset value	Description
TIMx_SR	0x10	RC_W0	0x0000_0000	TIM6 and TIM7 status register

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

TIM6 and TIM7 status register(TIMx_SR)Bit Description

Bit	Access	Description
[31:1]	R	Reserved, must be kept at reset value.

[0]	RC_W0	**UIF:** Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update occurred. 1: Update interrupt pending. This bit is set by hardware when the registers are updated: • At overflow or underflow and if UDIS = 0 in the TIMx_CR1 register. • When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register

16.3.10 TIM6 and TIM7 event generation register(TIMx_EGR, x = 6,7)

Register	Address offset	Access	Reset value	Description
TIMx_EGR	0x14	W	0x0000_0000	TIM6 and TIM7 event generation register

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

TIM6 and TIM7 event generation register(TIMx_EGR)Bit Description

Bit	Access	Description
[31:1]	R	Reserved, must be kept at reset value
[0]	W	**UG:** Update generation This bit can be set by software, it is automatically cleared by hardware. 0: No action. 1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected)

16.3.11 TIM6 and TIM7 counter (TIMx_CNT, x = 6,7)

Register	Address offset	Access	Reset value	Description
TIMx_CNT	0x24	RW	0x0000_0000	TIM6 and TIM7 counter

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

TIMx 计数器 (TIMx_CNT)Bit Description

Bit	Access	Description

[31:16]	R	Reserved
[15:0]	RW	**CNT[15:0]:** Counter value

16.3.12 TIM6 and TIM7 prescaler (TIMx_PSC, x = 6,7)

Register	Address offset	Access	Reset value	Description
TIMx_PSC	0x28	RW	0x0000_0000	TIM6 and TIM7 prescaler

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

TIM6 and TIM7 prescaler (TIMx_PSC)Bit Description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**PSC[15:0]:** Prescaler value The counter clock frequency CK_CNT is equal to f_{CK_PSC}/ (PSC[15:0] + 1). PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

16.3.13 TIM6 and TIM7 auto-reload register(TIMx_ARR, x = 6,7)

Register	Address offset	Access	Reset value	Description
TIMx_ARR	0x2C	RW	0x0000_0000	TIM6 and TIM7 auto-reload register

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

TIM6 and TIM7 auto-reload register(TIMx_ARR)Bit Description

Bit	Access	Description
[31:0]	RW	**ARR[31:0]:** Auto-reload value ARR is the value to be loaded into the actual auto-reload register. Refer to Section 16.3.1: Time-base unit for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.

17 Real-time clock(RTC)

17.1 RTC introduction

The real-time clock is an independent timer. The RTC provides a set of continuously running counters which can be used, with suitable software, to provide a clock-calendar function. The counter values can be written to set the current time/date of the system.

The RTC core and clock configuration (RCC_BDCR register) are in the Backup domain, which means that RTC setting and time are kept after reset or wakeup from Standby mode.

After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access. To enable access to the Backup registers and the RTC, proceed as follows:

- enable the power and backup interface clocks by setting the PWREN and BKPEN bits in the RCC_APB1ENR register
- set the DBP bit the Power Control Register (PWR_CR) to enable access to the Backup registers and RTC.

17.2 RTC main features

- Programmable prescaler: division factor up to2^{20}
- 32-bit programmable counter for long-term measurement
- Two separate clocks: PCLK1 for the APB1 interface and RTC clock (must be at least four times slower than the PCLK1 clock)
- The RTC clock source could be any of the following ones:
  1. HSE clock divided by 128
  2. LSE oscillator clock
  3. LSI oscillator clock
- Two separate reset types
  1. The APB1 interface is reset by system reset
  2. The RTC Core (Prescaler, Alarm, Counter and Divider) is reset only by a Backup domain reset
- Three dedicated maskable interrupt lines
  1. Alarm interrupt, for generating a software programmable alarm interrupt.
  2. Seconds interrupt, for generating a periodic interrupt signal with a programmable period length (up to 1 second).
  3. Overflow interrupt, to detect when the internal programmable counter rolls over to 0.

17.3 RTC functional description

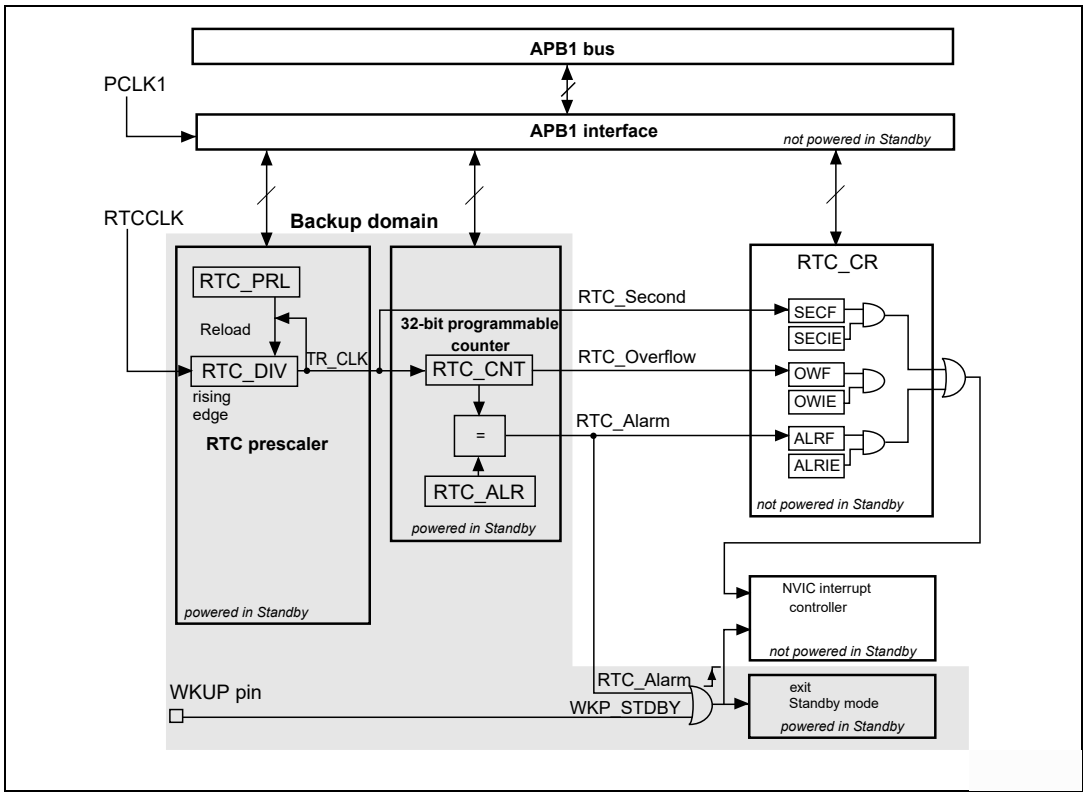
17.3.1 Overview

The RTC consists of two main units (see Figure 17.3-1). The first one (APB1 Interface) is used to interface with the APB1 bus. This unit also contains a set of 16-bit registers accessible from the APB1 bus in read or write mode (for more information refer to Section 17.4-1: RTC registers). The APB1 interface

is clocked by the APB1 bus clock in order to interface with the APB1 bus.

The other unit (RTC Core) consists of a chain of programmable counters made of two main blocks. The first block is the RTC prescaler block, which generates the RTC time base TR_CLK that can be programmed to have a period of up to 1 second. It includes a 20-bit programmable divider (RTC Prescaler). Every TR_CLK period, the RTC generates an interrupt (Second Interrupt) if it is enabled in the RTC_CR register. The second block is a 32-bit programmable counter that can be initialized to the current system time. The system time is incremented at the TR_CLK rate and compared with a programmable date (stored in the RTC_ALR register) in order to generate an alarm interrupt, if enabled in the RTC_CR control register.

Fig 17.3-1 RTC block diagram



17.3.2 Resetting RTC registers

All system registers are asynchronously reset by a System Reset or Power Reset, except for RTC_PRL, RTC_ALR, RTC_CNT, and RTC_DIV.

The RTC_PRL, RTC_ALR, RTC_CNT, and RTC_DIV registers are reset only by a Backup Domain reset.

17.3.3 Reading RTC registers

The RTC core is completely independent from the RTC APB1 interface.

Software accesses the RTC prescaler, counter and alarm values through the APB1 interface but the associated readable registers are internally updated at each rising edge of the RTC clock resynchronized by the RTC APB1 clock. This is also true for the RTC flags.

This means that the first read to the RTC APB1 registers may be corrupted (generally read as 0) if the APB1 interface has previously been disabled and the read occurs immediately after the APB1 interface is enabled but before the first internal update of the registers. This can occur if:

- A system reset or power reset has occurred
- The MCU has just woken up from Standby mode
- The MCU has just woken up from Stop mode

In all the above cases, the RTC core has been kept running while the APB1 interface was disabled (reset, not clocked or unpowered).
Consequently when reading the RTC registers, after having disabled the RTC APB1 interface, the software must first wait for the RSF bit (Register Synchronized Flag) in the RTC_CRL register to be set by hardware.

Note: *that the RTC APB1 interface is not affected by WFI and WFE low-power modes*

17.3.4 Configuring RTC registers

To write in the RTC_PRL, RTC_CNT, RTC_ALR registers, the peripheral must enter Configuration Mode. This is done by setting the CNF bit in the RTC_CRL register。
In addition, writing to any RTC register is only enabled if the previous write operation is finished. To enable the software to detect this situation, the RTOFF status bit is provided in the RTC_CR register to indicate that an update of the registers is in progress. A new value can be written to the RTC registers only when the RTOFF status bit value is ’1’.

Configuration procedure:

1. Poll RTOFF, wait until its value goes to ‘1’
2. Set the CNF bit to enter configuration mode
3. Write to one or more RTC registers
4. Clear the CNF bit to exit configuration mode
5. Poll RTOFF, wait until its value goes to ‘1’ to check the end of the write operation

Note: *The write operation only executes when the CNF bit is cleared; it takes at least three RTCCLK cycles to complete.*

17.3.5 RTC flag assertion

The RTC Second flag (SECF) is asserted on each RTC Core clock cycle before the update of the RTC Counter.
The RTC Overflow flag (OWF) is asserted on the last RTC Core clock cycle before the counter reaches 0x0000.
The RTC_Alarm and RTC Alarm flag (ALRF) (see Figure 17.3-2) are asserted on the last RTC Core clock cycle before the counter reaches the RTC Alarm value stored in the Alarm register increased by one (RTC_ALR + 1). The write operation in the RTC Alarm and RTC Second flag must be synchronized by using one of the following sequences:

The write operation in the RTC Alarm and RTC Second flag must be synchronized by using one of the following sequences:

- Use the RTC Alarm interrupt and inside the RTC interrupt routine, the RTC Alarm and/or RTC Counter registers are updated
- Wait for SECF bit to be set in the RTC Control register. Update the RTC Alarm and/or the RTC Counter register.

Fig 17.3-2 RTC second and alarm waveform example with PR=0x03, ALR=0x2F

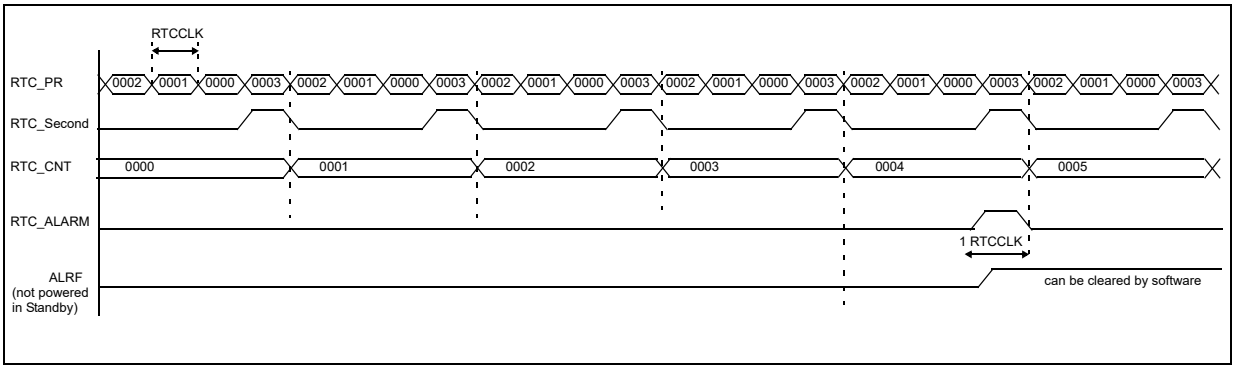
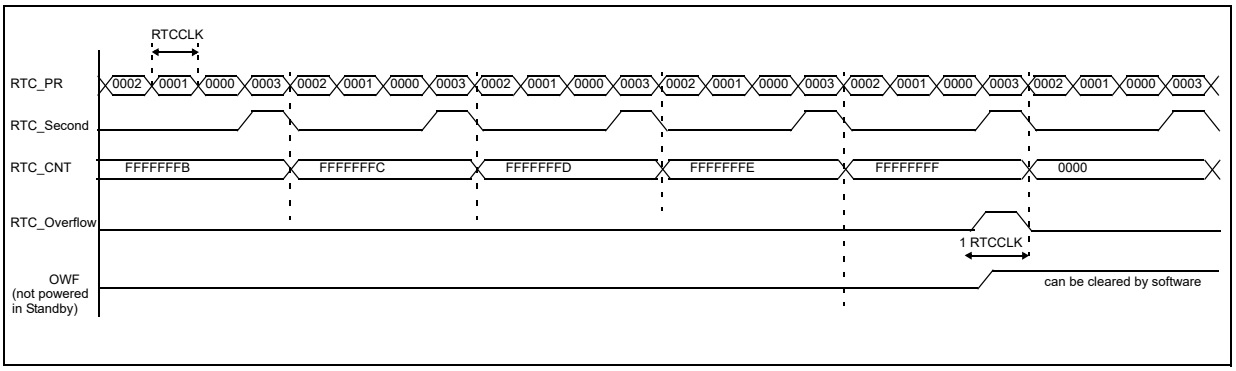


Fig 17.3-3 RTC Overflow waveform example with PR=0003



17.4 RTC registers

Tab 17.4-1 RTC registers map

offset	Register	Reset value	Description
RTC base address: RTC_BA = 0x4000_2800			
0x00	RTC_CRH	0x0000_0000	RTC control register high
0x04	RTC_CRL	0x0000_0020	RTC control register low
0x08	RTC_PRLH	0x0000_0000	RTC prescaler high load register
0x0C	RTC_PRLL	0x0000_0000	RTC prescaler low load register
0x10	RTC_DIVH	0x0000_0000	RTC prescaler high divider register
0x14	RTC_DIVL	0x0000_8000	RTC prescaler low divider register
0x18	RTC_CNTH	0x0000_0000	RTC high counter register
0x1C	RTC_CNTL	0x0000_0000	RTC low counter register
0x20	RTC_ALRH	0x0000_0000	RTC high alarm register
0x24	RTC_ALRL	0x0000_0000	RTC low alarm register

17.4.1 RTC control register high(RTC_CRH)

Note: *These bits are used to mask interrupt requests. Note that at reset all interrupts are disabled, so it is possible to write to the RTC registers to ensure that no interrupt requests are pending after initialization. It is not possible to write to the RTC_CRH register when the peripheral is completing a previous write operation（flagged by RTOFF=0, see Section 17.3.4）.*

The RTC functions are controlled by this control register. Some bits must be written using a specific configuration procedure

Register	Address offset	Access	Reset value	Description
RTC_CRH	0x00	RW	0x0000_0000	RTC control register high

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

RTC control register high(RTC_CRH)bit description

Bit	Access	Description
[31:3]	R	Reserved
[2]	RW	**OWIE:** Overflow interrupt enable 0: Overflow interrupt is masked. 1: Overflow interrupt is enabled.
[1]	RW	**ALRIE:** Alarm interrupt enable 0: Alarm interrupt is masked. 1: Alarm interrupt is enabled
[0]	RW	**SECIE:** Second interrupt enable 0: Second interrupt is masked. 1: Second interrupt is enabled

17.4.2 RTC control register low(RTC_CRL)

Register	Address offset	Access	Reset value	Description
RTC_CRL	0x04	RW	0x0000_0020	RTC control register low

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

RTC control register low(RTC_CRL)bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	R	**RTOFF:** RTC operation OFF With this bit the RTC reports the status of the last write operation performed on its registers, indicating if it has been completed or not. If its value is ‘0’ then it is not possible to write to any of the RTC registers. This bit is read only. 0: Last write operation on RTC registers is still ongoing 1: Last write operation on RTC registers terminated
[4]	RW	**CNF:** Configuration flag This bit must be set by software to enter in configuration mode so as to allow new values to be written in the RTC_CNT, RTC_ALR or RTC_PRL registers. The write operation is only executed when the CNF bit is reset by software after has been set. 0: Exit configuration mode (start update of RTC registers) 1: Enter configuration mode
[3]	RC_W0	**RSF:** Registers synchronized flag This bit is set by hardware at each time the RTC_CNT and RTC_DIV registers are updated and cleared by software. Before any read operation after an APB1 reset or an APB1 clock stop, this bit must be cleared by software, and the user application must wait until it is set to be sure that the RTC_CNT, RTC_ALR or RTC_PRL registers are synchronized. 0: Registers not yet synchronized. 1: Registers synchronized.
[2]	RC_W0	**OWF:** Overflow flag This bit is set by hardware when the 32-bit programmable counter overflows. An interrupt is generated if OWIE=1 in the RTC_CRH register. It can be cleared only by software. Writing ‘1’ has no effect 0: Overflow not detected 1: 32-bit programmable counter overflow occurred
[1]	RC_W0	**ALRF:** Alarm flag This bit is set by hardware when the 32-bit programmable counter reaches the threshold set in the RTC_ALR register. An interrupt is generated if ALRIE=1 in the RTC_CRH register. It can be cleared only by software. Writing ‘1’ has no effect. 0: Alarm not detected 1: Alarm detected

[0]	RC_W0	**SECF:** Second flag This bit is set by hardware when the 32-bit programmable prescaler overflows, thus incrementing the RTC counter. Hence this flag provides a periodic signal with a period corresponding to the resolution programmed for the RTC counter (usually one second). An interrupt is generated if SECIE=1 in the RTC_CRH register. It can be cleared only by software. Writing ‘1’ has no effect. 0: Second flag condition not met. 1: Second flag condition met.

The functions of the RTC are controlled by this control register. It is not possible to write to the RTC_CR register while the peripheral is completing a previous write operation (flagged by RTOFF=0,Configuring RTC registers).

Note:

1. *Any flag remains pending until the appropriate RTC_CR request bit is reset by software,indicating that the interrupt request has been granted.*
2. *At reset the interrupts are disabled, no interrupt requests are pending and it is possible to write to the RTC registers.*
3. *The OWF, ALRF, SECF and RSF bits are not updated when the APB1 clock is not running.*
4. *The OWF, ALRF, SECF and RSF bits can only be set by hardware and only cleared by software.*
5. *If ALRF = 1 and ALRIE = 1, the RTC global interrupt is enabled. If EXTI Line 17 is also enabled through the EXTI Controller, both the RTC global interrupt and the RTC Alarm interrupt are enabled.*
6. *If ALRF = 1, the RTC Alarm interrupt is enabled if EXTI Line 17 is enabled through the EXTI Controller in interrupt mode. When the EXTI Line 17 is enabled in event mode, a pulse is generated on this line (no RTC Alarm interrupt generation).*

17.4.3 RTC prescaler load register(RTC_PRLH/RTC_PRLL)

The Prescaler Load registers keep the period counting value of the RTC prescaler. They are write-protected by the RTOFF bit in the RTC_CR register, and a write operation is allowed if the RTOFF value is ‘1’.

Write only (see Section 17.3.4: Configuring RTC registers)

RTC prescaler load register high (RTC_PRLH)

Register	Address offset	Access	Reset value	Description
RTC_PRLH	0x08	W	0x0000_0000	RTC prescaler load register high

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

RTC prescaler load register high(RTC_PRLH)bit description

Bit	Access	Description
[31:4]	R	Reserved

[3:0]	W	**PRL[19:16]:** RTC prescaler reload value high These bits are used to define the counter clock frequency according to the following formula: $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$

RTC prescaler load register low(RTC_PRLL)

Register	Address offset	Access	Reset value	Description
RTC_PRLL	0x0C	W	0x0000_0000	RTC prescaler load register low

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

RTC prescaler load register low(RTC_PRLL)bit description

Bit	Access	Description
[31:15]	R	Reserved
[15:0]	W	**PRL[15:0]:** RTC prescaler reload value low These bits are used to define the counter clock frequency according to the following formula: $f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$ **Caution:** *The zero value is not recommended. RTC interrupts and flags cannot be asserted correctly.*

Note: *If the input clock frequency(RTC clock)is 32.768 kHz, write 0x7FFF in this register to get a signal period of 1 second.*

17.4.4 RTC prescaler divider register (RTC_DIVH / RTC_DIVL)

During each period of TR_CLK, the counter inside the RTC prescaler is reloaded with the value stored in the RTC_PRL register. To get an accurate time measurement it is possible to read the current value of the prescaler counter, stored in the RTC_DIV register, without stopping it. This register is read-only and it is reloaded by hardware after any change in the RTC_PRL or RTC_CNT registers.

RTC prescaler divider register high(RTC_DIVH)

Register	Address offset	Access	Reset value	Description
RTC_DIVH	0x10	R	0x0000_0000	RTC prescaler divider register high

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

RTC prescaler divider register high(RTC_DIVH)bit description

Bit	Access	Description
[31:4]	R	Reserved
[3:0]	R	**DIV[19:16]:** RTC clock divider high

RTC prescaler divider register low(RTC_DIVL)

Register	Address offset	Access	Reset value	Description
RTC_DIVL	0x14	R	0x0000_8000	RTC prescaler divider register low

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

RTC prescaler divider register low(RTC_DIVL)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	R	**DIV[15:0]:** RTC clock divider low

17.4.5 RTC counter register (RTC_CNTH / RTC_CNTL)

The RTC core has one 32-bit programmable counter, accessed through two 16-bit registers; the count rate is based on the TR_CLK time reference, generated by the prescaler.

RTC_CNT registers keep the counting value of this counter. They are write-protected by bit RTOFF in the RTC_CR register, and a write operation is allowed if the RTOFF value is ‘1’. A write operation on the upper (RTC_CNTH) or lower (RTC_CNTL) registers directly loads the corresponding programmable counter and reloads the RTC Prescaler. When reading, the current value in the counter (system date) is returned.

RTC counter register high(RTC_CNTH)

Register	Address offset	Access	Reset value	Description
RTC_CNTH	0x18	RW	0x0000_0000	RTC counter register high

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

RTC counter register high(RTC_CNTH)bit description

Bit	Access	Description
[31:15]	R	Reserved
[15:0]	RW	**CNT[31:16]:** RTC counter high Reading the RTC_CNTH register, the current value of the high part of the RTC Counter register is returned. To write to this register it is necessary to enter configuration mode(see Section：17.3.4)

RTC counter register low(RTC_CNTL)

Register	Address offset	Access	Reset value	Description
RTC_CNTL	0x1C	RW	0x0000_0000	RTC counter register low

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

RTC counter register low(RTC_CNTL)bit description

Bit	Access	Description
[31:15]	R	Reserved
[15:0]	RW	**CNT[15:0]:** RTC counter low Reading the RTC_CNTL register, the current value of the lower part of the RTC Counter register is returned. To write to this register it is necessary to enter configuration mode(see Section：17.3.4)

17.4.6 RTC alarm register high (RTC_ALRH / RTC_ALRL)

When the programmable counter reaches the 32-bit value stored in the RTC_ALR register, an alarm is triggered and the RTC_alarmIT interrupt request is generated. This register is write-protected by the RTOFF bit in the RTC_CR register, and a write operation is allowed if the RTOFF value is ‘1’.

RTC alarm register high(RTC_ALRH)

Register	Address offset	Access	Reset value	Description
RTC_ALRH	0x20	W	0x0000_0000	RTC alarm register high

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

RTC alarm register high(RTC_ALRH)bit description

Bit	Access	Description
[31:15]	R	Reserved
[15:0]	W	**ALR[31:16]:** RTC alarm high The high part of the alarm time is written by software in this register. To write to this register it is necessary to enter configuration mode(see Section：17.3.4)

RTC alarm register low(RTC_ALRL)

Register	Address offset	Access	Reset value	Description
RTC_ALRL	0x24	W	0x0000_0000	RTC alarm register low

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

RTC alarm register low(RTC_ALRL)bit description

Bit	Access	Description
[31:15]	R	Reserved
[15:0]	W	**ALR[15:0]:** RTC alarm low The low part of the alarm time is written by software in this register. To write to this register it is necessary to enter configuration mode(see Section：17.3.4)

18 Independent watchdog (IWDG)

18.1 IWDG introduction

The devices have two embedded watchdog peripherals which offer a combination of high safety level, timing accuracy and flexibility of use. Both watchdog peripherals (Independent and Window) serve to detect and resolve malfunctions due to software failure, and to trigger system reset or an interrupt (window watchdog only) when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails. The window watchdog (WWDG) clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. The WWDG is best suited to applications which require the watchdog to react within an accurate timing window. For further information on the window watchdog, refer to Section：19.

18.2 IWDG main features

- Free-running downcounter
- clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x000 is reached

18.3 IWDG functional description

Figure: 18.3-1 shows the functional blocks of the independent watchdog module

When the independent watchdog is started by writing the value 0xCCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0xAAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

18.3.1 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and will generate a reset unless the Key register is written by the software before the counter reaches end of count.

18.3.2 Register access protection

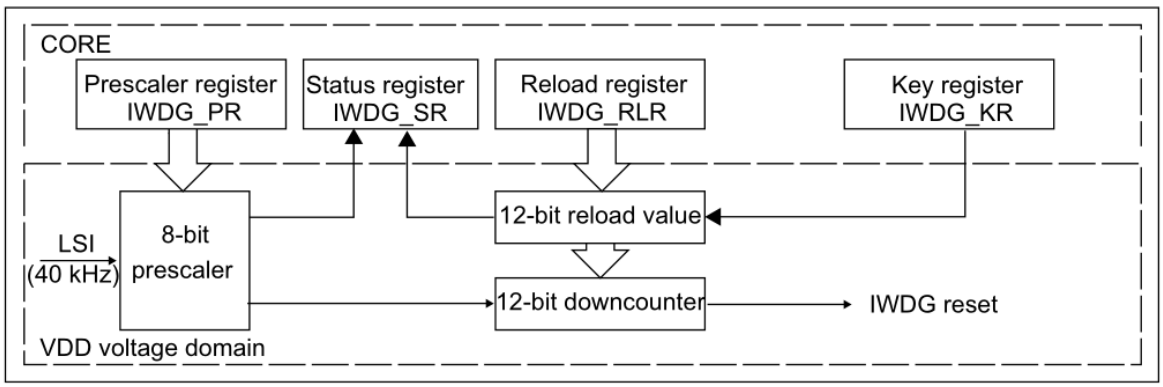
Write access to the IWDG_PR and IWDG_RLR registers is protected. To modify them, first write the code 0x5555 in the IWDG_KR register. A write access to this register with a different value will break the sequence and register access will be protected again. This implies that it is the case of the reload operation (writing 0xAAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value is on going.

18.3.3 Debug mode

When the microcontroller enters debug mode (Cortex-M3 core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module. For more details, refer to 30.5.2

Fig 18.3-1 Independent watchdog block diagram



Note: *The watchdog function is implemented in the VDD voltage domain, still functional in Stop and Standby modes.*

Tab 18.3-1 Min/max IWDG timeout period (in ms) at 40 kHz (LSI)

Prescaler divider	PR[2:0] bits	Min timeout RL[11:0] = 0x000	Max timeout RL[11:0] = 0xFFF
/4	0	0.1	409.6
/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(or 7)	6.4	26214.4

Note: *These timings are given for a 40 kHz clock but the microcontroller internal RC frequency can vary. Refer to the LSI oscillator characteristics table in the device datasheet for maximum and minimum values.*

The LSI can be calibrated so as to compute the IWDG timeout with an acceptable accuracy. For more details refer to Section : 7.3.5 (LSI clock).

18.4 IWDG Register

Tab 18.4-1 IWDG register map

Offset	Register	Reset value	Description
IWDG_BA = 0x4000_3000			
0x00	IWDG_KR	0x0000_0000	IWDG Key register (reset by Standby mode)
0x04	IWDG_PR	0x0000_0000	IWDG Prescaler register
0x08	IWDG_RLR	0x0000_0000	IWDG Reload register (reset by Standby mode)
0x0C	IWDG_SR	0x0000_0000	IWDG Status register (not reset by Standby mode)

18.4.1 IWDG Key register (IWDG_KR)

Register	Address offset	Access	Reset value	Description
IWDG_KR	0x00	W	0x0000_0000	IWDG Key register (reset by Standby mode)

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

IWDG Key register(IWDG_KR)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value
[15:0]	W	**KEY[15:0]:** Key value (write only, read 0000h) These bits must be written by software at regular intervals with the key value AAAAh, otherwise the watchdog generates a reset when the counter reaches 0. Writing the key value 5555h to enable access to the IWDG_PR and IWDG_RLR registers (see Section 18.3.2) Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

18.4.2 IWDG Prescaler register (IWDG_PR)

Register	Address offset	Access	Reset value	Description
IWDG_PR	0x04	RW	0x0000_0000	IWDG Prescaler register

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

IWDG Prescaler register(IWDG_PR)bit description

Bit	Access	Description
[31:3]	-	Reserved, must be kept at reset value.
[2:0]	RW	**PR[2:0]:** Prescaler divider These bits are write access protected see Section 18.3.2. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider. 000: divider/4 001: divider/8 010: divider/16 011: divider/32 100: divider/64 101: divider/128 110: divider/256 111: divider/256 **Note:** *Reading this register returns the prescaler value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG_SR register is reset.*

18.4.3 IWDG Reload register (IWDG_RLR)

Register	Address offset	Access	Reset value	Description
IWDG_RLR	0x08	RW	0x0000_0000	IWDG Reload register (reset by Standby mode)

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

IWDG Reload register(IWDG_RLR)bit description

Bit	Access	Description
[31:12]	R	Reserved, must be kept at reset value.
[11:0]	RW	**RL[11:0]:** Watchdog counter reload value These bits are write access protected see Section 18.3.2. They are written by software to define the value to be loaded in the watchdog counter each time the value AAAAh is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to Table 18.3-1. The RVU bit in the IWDG_SR register must be reset in order to be able to change the reload value. **Note:** *Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on this register. For this reason the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset*

18.4.4 IWDG Status register (IWDG_SR)

Register	Address offset	Access	Reset value	Description
IWDG_SR	0x0C	RW	0x0000_0000	IWDG Status register(not reset by Standby mode)

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

IWDG Status register(IWDG_SR)bit description

Bit	Access	Description
[31:2]	R	Reserved, must be kept at reset value.
[1]	R	**RVU:** Watchdog counter reload value update This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the VDD voltage domain (takes up to 5 RC 40 kHz cycles). Reload value can be updated only when RVU bit is reset.
[0]	R	**PVU:** Watchdog prescaler value update This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the VDD voltage domain (takes up to 5 RC 40 kHz cycles). Prescaler value can be updated only when PVU bit is reset.

Note:

If several reload values or prescaler values are used by application, it is mandatory to wait until RVU bit is reset before changing the reload value and to wait until PVU bit is reset before changing the prescaler value. However, after updating the prescaler and/or the reload value it is not necessary to wait until RVU or PVU is reset before continuing code execution (even in case of low-power mode entry, the write operation is taken into account and will complete)

19 Window watchdog (WWDG)

19.1 WWDG introduction

The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

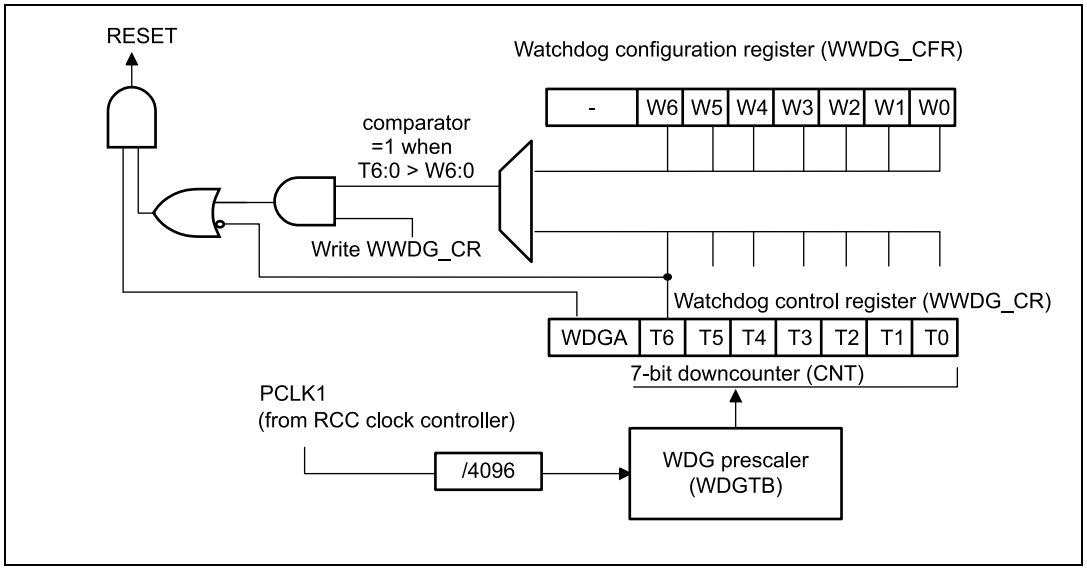
19.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 0x40
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see Figure 184)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

19.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) rolls over from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

Fig 19.3-1 Watchdog block diagram



The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

Controlling the downcounter

This downcounter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see Figure 19.3-2). The Configuration register (WWDG_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F. Figure 19.3-2 describes the window watchdog process.

Note: *The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared)*

Advanced watchdog interrupt feature

The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

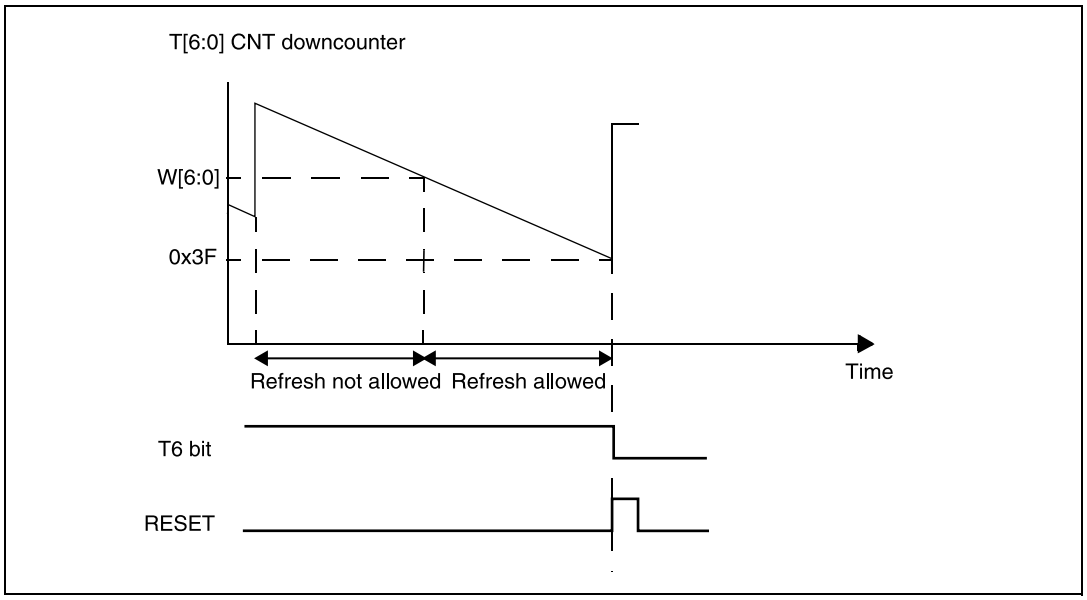
The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

Note: *When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset will eventually be generated.*

19.3.1 How to program the watchdog timeout

Note: *When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.*

Fig 19.3-2 Window watchdog timing diagram



The formula to calculate the WWDG timeout value is given by:

$$t_{WWDG} = t_{PCLK1} \times 4096 \times 2^{WDGTB[1:0]} \times (T[5:0] + 1) \quad (ms) \tag{19.1}$$

where :
t_{WWDG} : WWDG timeout
t_{PCLK1}: APB1 clock period measured in ms
4096: value corresponding to internal divider

As an example, let us assume APB1 frequency is equal to 48 MHz, WDGTB[1:0] is set to 3 and T[5:0] is set to 63:

$$t_{WWDG} = 1/48000 \times 4096 \times 2^3 \times (63 + 1) = 43.69ms \tag{19.2}$$

Refer to Table 19.3-1 for the minimum and maximum values of the t_{WWDG}

Tab 19.3-1 Minimum and maximum timeout values @48 MHz (f_{PCLK1})

Prescaler	WDGTB	Min timeout value	Max timeout value
1	0	85 µs	5.46 ms
2	1	170 µs	10.92 ms
4	2	341 µs	21.84 ms
5	3	682 µs	43.69 ms

19.3.2 Debug mode

When the microcontroller enters debug mode (Cortex-M3 core halted), the WWDG counter either continues to work normally or stops, depending on DBG_WWDG_STOP configuration bit in DBG module. For more details, refer to Section 30.5.2

19.4 WWDG Register

Tab 19.4-1 WWDG register map

offset	Register	Reset value	Description
WWDG_BA = 0x4000_2C00			
0x00	WWDG_CR	0x0000_007F	WWDG Control register
0x04	IWDG_CFR	0x0000_007F	WWDG Configuration Register
0x08	WWDG_SR	0x0000_0000	WWDG Status Register

Note: *The peripheral registers have to be accessed by half-words (16 bits) or words (32 bits).*

19.4.1 WWDG Control register (WWDG_CR)

Register	Address offset	Access	Reset value	Description
WWDG_CR	0x00	RW	0x0000_007F	WWDG Control register

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

WWDG Control register(WWDG_CR)bit description

Bit	Access	Description
[31:8]	-	Reserved, must be kept at reset value.
[7]	RS	**WDGA:** Activation bit This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset. 0: Watchdog disabled 1: Watchdog enabled
[6:0]	RW	**T[6:0]:** 7-bit counter (MSB to LSB) These bits contain the value of the watchdog counter. It is decremented every (4096 x $2^{WDGTB[1:0]}$) PCLK1 cycles. A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

19.4.2 WWDG Configuration register (WWDG_CFR)

Register	Address offset	Access	Reset value	Description
WWDG_CFR	0x04	RW	0x0000_007F	WWDG Configuration register

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

WWDG Configuration register (WWDG_CFR)bit description

Bit	Access	Description
[31:10]	R	Reserved, must be kept at reset value.
[9]	RS	**EWI:** Early wakeup interrupt When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.
[8:7]	RW	**WDGTB[1:0]:** Timer base The time base of the prescaler can be modified as follows: 00: CK Counter Clock (PCLK1 div 4096) div 1 01: CK Counter Clock (PCLK1 div 4096) div 2 10: CK Counter Clock (PCLK1 div 4096) div 4 11: CK Counter Clock (PCLK1 div 4096) div 8
[6:0]	W	**W[6:0]:** 7-bit window value These bits contain the window value to be compared to the downcounter

19.4.3 WWDG Status register (WWDG_SR)

Register	Address offset	Access	Reset value	Description
WWDG_SR	0x08	RW	0x0000_0000	WWDG Status register

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

WWDG Status register(WWDG_SR)bit description

Bit	Access	Description
[31:1]	-	Reserved, must be kept at reset value.
[0]	RC_W0	**EWIF:** Early wakeup interrupt flag This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing ‘0’. A write of ‘1’ has no effect. This bit is also set if the interrupt is not enabled.

20 Controller area network (bxCAN)

20.1 bxCAN introduction

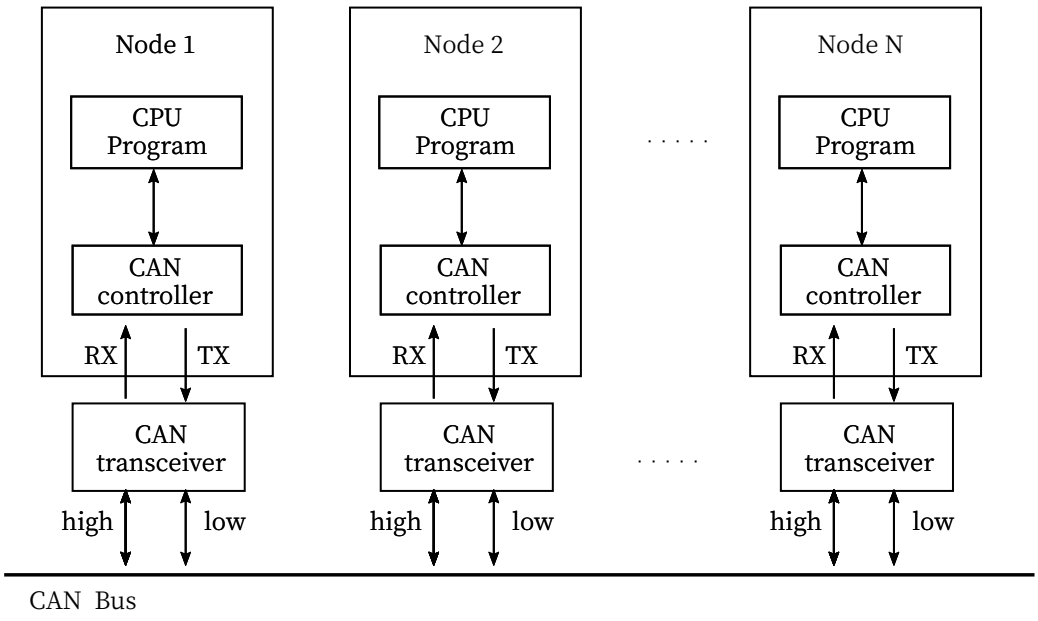
CAN (Controller Area Network), is a serial communication network capable of realizing distributed real-time control. It has many advantages such as high-speed short distance, low-speed long distance, lossless bit arbitration mechanism, multi-master device structure and etc. It is widely used in automobiles, ships, robots, automatic control, etc.

20.2 bxCAN main features

- Compliant with CAN protocol 2.0A/2.0B and ISO 11898-1: 2015 standard
- Support standard frame and extended frame structure
- Support legacy and FD frames
- Support receiving filter (single/dual mode)
- Transceiver with a 128-byte FIFO (RX_MEM/TX_MEM)
- Support sending and receiving delay compensation
- Support normal and listen-only modes
- Selectable 7 interrupt sources

20.3 General Description

When the CPU executes the application program, it will call the CAN driver layer software to control the CAN controller. The CAN controller will automatically complete the task of sending and receiving frames. Filters, sending and receiving FIFOs and multiple interrupt sources can effectively reduce the CPU load



20.3.1 Kernel

The controller can receive and send CAN messages fully automatically; and fully supports standard frame identifiers (11 bits) and extended frame identifiers (29 bits);

20.3.2 Control, observation registers

The driver can use these registers to:

- Configure parameters such as baud rate
- Configure sending messages
- Request to send
- Receive telegrams with specified identifiers
- Handle interrupts
- Handle receive messages

20.4 Control and observation registers

20.4.1 Sending FIFO

Filling through the TXBUF interface each time, only one message filled in each round.

20.4.2 Receiving Filter

The two sets of filters share 32-bit matching parameters, and the driver can be configured as required, and the unnecessary messages are directly ignored.

20.4.3 Receiving FIFO

The received message will be written into the FIFO according to the specified format, and the driver can read it through the RXBUF interface. After reading a message, the RMC needs to be reduced by one.

20.4.4 Working Mode

CAN controller has 5 working modes: reset, initialization, normal, silent and test mode. After the hardware reset, the controller works in the reset mode, and will not do any active operation at this time, so the driver can configure each parameter register. After the configuration, one of the normal, silent and test modes will be selected and then the reset mode will be exited. After exiting the reset mode, the controller will not immediately start sending and receiving operations, but will first enter the initialization mode. At this time, the controller will monitor the bus until 11 consecutive recessive bits before exiting, and enter the next mode according to the previous configuration.

- Reset Mode: This mode is mainly used to configure various working parameter registers. In order to prevent misuse, these registers cannot be modified after exiting the reset mode.
- Initialization Mode: In order to prevent misjudgment of the frame header, after exiting the reset mode, the controller will wait in the initialization mode until it receives 11 recessive bits continuously before entering the next mode.

- Normal Mode: After entering this mode, the controller will normally receive or send the bus.
- Silent Mode: This mode is mainly used for bus speed detection or analysis. At this time, the controller will receive the bus signal and will not perform any sending operations on the bus. Even if it is successfully received, it will not send ACK, and the error counter will not be updated.
- Test Mode: This mode is mainly used for format debugging. At this time, the controller will close the reception of the bus signal, and at the same time directly return the sending signal to the receiving end. You can choose whether the sending signal enters the bus or not through configuration.

20.5 Functions Description

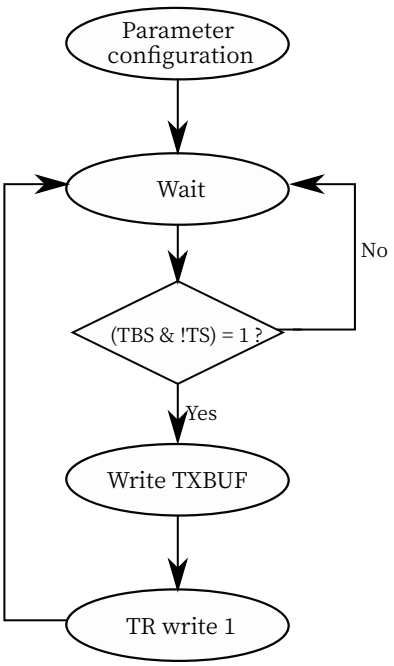
20.5.1 Sending Function

The process of sending a message :

1. Fill in the information of the entire message through the TXBUF interface: identifier, data length, data and etc;
2. when things are ready, only write 1 to TR to start the sending process;

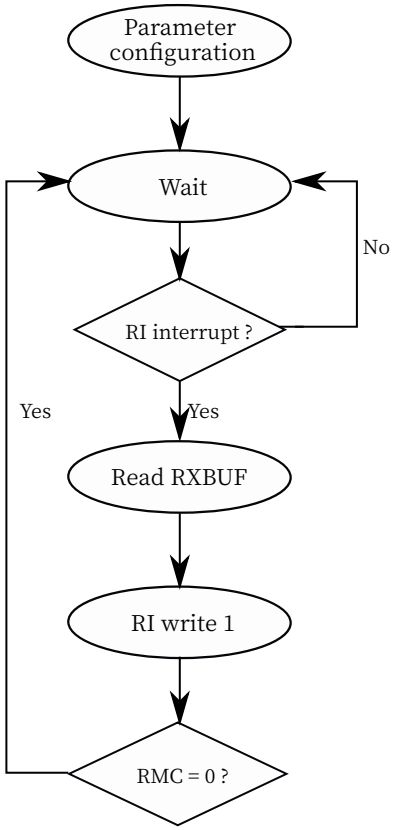
Suspense:
After the function comes into effect, messages can be stopped sending by writing 1 to AT , but whether it can be stopped depends on the state machine, if it has already started sending, it cannot be stopped

Automatic repeat mode is prohibited:
If it is necessary to turn off the automatic repeat function of this transmission, writing 1 to TR and AT at the same time can perform a single non-repeat transmission, or first setting the DAR to be 1 and then writing TR, so that it will not be repeated every time;



20.5.2 Receiving function

The process of receiving messages :
After the parameters are configured, it is necessary to wait for the RI interrupt, and take the messages from RXBUF sequentially according to the interrupts. 1 is written to RI every time a message is fetched;



20.5.3 Acceptance Filter (ACF)

When the receiving end successfully receives new messages from the bus, not all of them will be saved, but will be checked by one or two receiving filters first, which is also called identifier checking. The messages that meet the checking conditions will be saved and written to RX_MEM. The way of identifier checking is mainly selected through 2 registers: AMR and ACR.

Each identifier will have a corresponding AMR and ACR register. When AMR=1, the bit is checked by default. When AMR=0, the identifier must be consistent with ACR to pass. Only when the result of each bit is passed will the check be considered as passed, allowing to write to RX_MEM:

IDx	AMRx	ACRx	Results
x	1	x	Check passed
0	0	0	Check passed
1	0	x	Check Failure
0	0	1	Check Failure
1	0	1	Check passed

In order to increase flexibility, 2 filters can be used to check in parallel. At this time, there will be two

sets of ID configurations. As long as one set of checks passes, it is allowed to write RX_MEM. The following is the corresponding matching data of each bit of AMR/ACR in different modes:

Tab 20.5-1 AMR/ACR corresponding matching data in different modes

Bit	single filter		Dual filter	
	Standard Frame	Extended Frame	Standard Frame	Extended Frame
31	Data 2 [7]	ID 4 bit	Group 2 ID 20 bit	Group 2 ID 20 bit
30	Data 2 [6]	ID 3 bit	Group 2 ID 19 bit	Group 2 ID 19 bit
29	Data 2 [5]	ID 2 bit	Group 2 ID 18 bit	Group 2 ID 18 bit
28	Data 2 [4]	ID 1 bit	Group 2 RTR bit	Group 2 ID 17 bit
27	Data 2 [3]	ID 0 bit	Group 1 [3]	Group 2 ID 16 bit
26	Data 2 [2]	RTR bit	Group 1 [2]	Group 2 ID 15 bit
25	Data 2 [1]	-	Group 1 [1]	Group 2 ID 14 bit
24	Data 2 [0]	-	Group 1 [0]	Group 2 ID 13 bit
23	Data 1 [7]	ID 12 bit	Group 2 ID 28 bit	Group 2 ID 28 bit
22	Data 1 [6]	ID 11 bit	Group 2 ID 27 bit	Group 2 ID 27 bit
21	Data 1 [5]	ID 10 bit	Group 2 ID 26 bit	Group 2 ID 26 bit
20	Data 1 [4]	ID 9 bit	Group 2 ID 25 bit	Group 2 ID 25 bit
19	Data 1 [3]	ID 8 bit	Group 2 ID 24 bit	Group 2 ID 24 bit
18	Data 1 [2]	ID 7 bit	Group 2 ID 23 bit	Group 2 ID 23 bit
17	Data 1 [1]	ID 6 bit	Group 2 ID 22 bit	Group 2 ID 22 bit
16	Data 1 [0]	ID 5 bit	Group 2 ID 21 bit	Group 2 ID 21 bit
15	ID 20 bit	ID 20 bit	Group 1 ID 20 bit	Group 1 ID 20 bit
14	ID 19 bit	ID 19 bit	Group 1 ID 19 bit	Group 1 ID 19 bit
13	ID 18 bit	ID 18 bit	Group 1 ID 18 bit	Group 1 ID 18 bit
12	RTR bit	ID 17 bit	Group 1 RTR bit	Group 1 ID 17 bit
11	-	ID 16 bit	Group 1 [7]	Group 1 ID 16 bit
10	-	ID 15 bit	Group 1 [6]	Group 1 ID 15 bit
9	-	ID 14 bit	Group 1 [5]	Group 1 ID 14 bit
8	-	ID 13 bit	Group 1 [4]	Group 1 ID 13 bit
7	ID 28 bit	ID 28 bit	Group 1 ID 28 bit	Group 1 ID 28 bit
6	ID 27 bit	ID 27 bit	Group 1 ID 27 bit	Group 1 ID 27 bit
5	ID 26 bit	ID 26 bit	Group 1 ID 26 bit	Group 1 ID 26 bit
4	ID 25 bit	ID 25 bit	Group 1 ID 25 bit	Group 1 ID 25 bit
3	ID 24 bit	ID 24 bit	Group 1 ID 24 bit	Group 1 ID 24 bit
2	ID 23 bit	ID 23 bit	Group 1 ID 23 bit	Group 1 ID 23 bit
1	ID 22 bit	ID 22 bit	Group 1 ID 22 bit	Group 1 ID 22 bit
0	ID 21 bit	ID 21 bit	Group 1 ID 21 bit	Group 1 ID 21 bit

20.5.4 TX_MEM and RX_MEM Layout

TX_MEM is used to store the information of a transmit frame; RX_MEM is used to store the information of all received frames.

Because the ID digits of the standard frame and the extended frame are different, their layout is slightly different. The following is the layout of the standard frame:

Address		Standard Frame								Description
0x0	7:0	FF	RTR	FDF	BRS	DLC3	DLC2	DLC1	DLC0	
	15:8	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	
	23:16	ID2	ID1	ID0	X/RTR	X/ESI	X/0	X/0	X/0	TX FIFO/RX FIFO
	31:24	data1								DLC greater than 0valid
0x1	7:0	data2								DLC greater than 1valid
	15:8	data3								DLC greater than 2valid
	23:16	data4								DLC greater than 3valid
	31:24	data5								DLC greater than 4valid
0x2	7:0	data6								DLC greater than 5valid
	15:8	data7								DLC greater than 6valid
	23:16	data8								DLC greater than 7valid
	31:24	data9								FD frame DLC greater than 8 valid FD frame DLC greater than 9valid
0x3	7:0	data10								
	15:8	data11								
	23:16	data12								
	31:24	data13								
…	7:0	…								
	15:8	…								
	23:16	…								
	31:24	…								
0x10	7:0	data62								FD frame DLC greater than 14 valid
	15:8	data63								
	23:16	data64								
	31:24									

Extended Frame 20.5-2，where each bit is defined as follows::

FF：frame format，0：standard frame，1：extended frame；
RTR：remote enable，0：Data frame,，1：remote frame；
FDF：FD enable,，0：legacy frame；1：FD frame；
BRS：FD frame dual rate switch, 0：Data same frequency for data andarbitration domains；1：different frequency used；
ESI：FD frame error mode,，0：active error state；1：passive error state；
X：arbitrary value!；
ID：：the identifier of the message；
DLC：data length code, the length of the padded data behind should match the DLC, the DLC must be matched with 0when sending remote frames；

Tab 20.5-2 Extended frame

Address		extended frame								Description
0x0	7:0	FF	RTR	FDF	BRS/0	DLC3	DLC2	DLC1	DLC0	TX_MEM/RX_MEM
	15:8	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	
	23:16	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13	
	31:24	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	
0x1	7:0	ID4	ID3	ID2	ID1	ID0	X/RTR	ESI/0	X/0	TX_MEM/RX_MEM
	15:8	data1								DLC greater than 0 valid
	23:16	data2								DLC greater than 1 valid
	31:24	data3								DLC greater than 2 valid
0x2	7:0	data4								DLC greater than 3 valid
	15:8	data5								DLC greater than 4 valid
	23:16	data6								DLC greater than 5 valid
	31:24	data7								DLC greater than 6 valid
0x3	7:0	data8								DLC greater than 7 valid
	15:8	data9								FD frame DLC greater than 8 valid
	23:16	data10								
	31:24	data11								
…	7:0	…								
	15:8	…								
	23:16	…								
	31:24	…								
0x10	7:0	data60								FD frame DLC greater than 14 valid
	15:8	data61								
	23:16	data62								
	31:24	data63								
0x11	7:0	data64								
	15:8									
	23:16									
	31:24									

20.5.5 DLC Code

Conventional frames will not exceed 8 in data length, but FD frames will exceed 8 in data length and will be encodedsomewhat differently in length:

DLC	Data Data length（in bytes）	
	Traditiona frames	FD frame
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	8	12
10	8	16
11	8	20
12	8	24
13	8	32
14	8	48
15	8	64

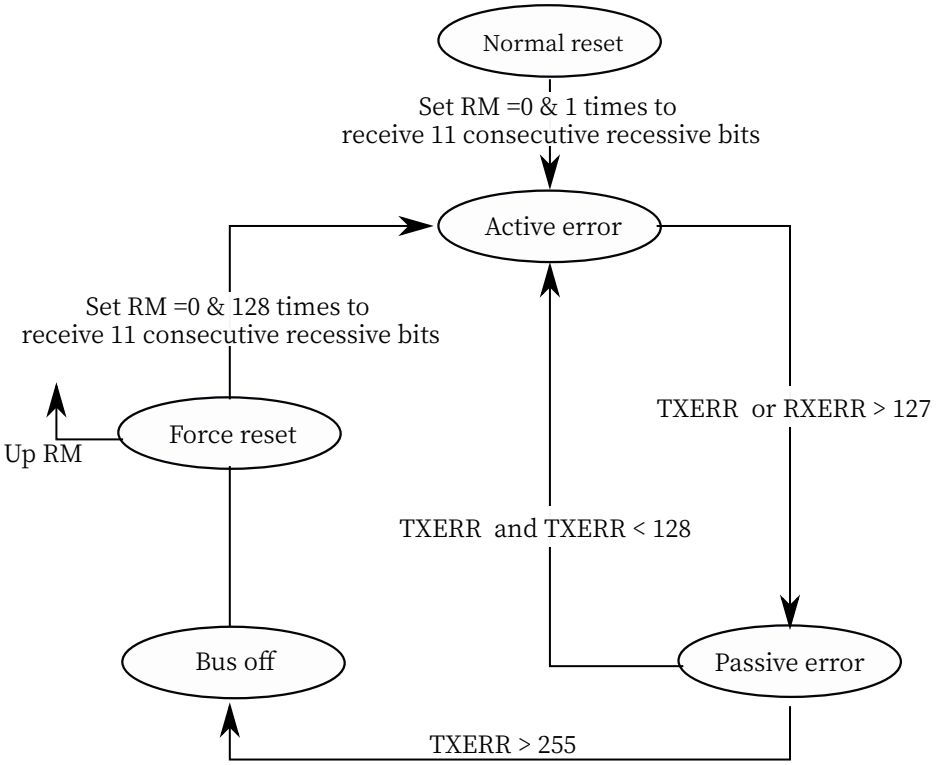
20.5.6 Error Management

There are 5 types of errors specified by CAN::

Name	Causes
Filling error	6 consecutive identical levels
Bit check error	Listening to the bus after sendingand finding inconsistencies
CRC check error	CRC sequences and calculationresults do not match
ACK detection error	ACK bit is implicit
Formatdetection error	Fixed format bits (e.g. boundarycharacters etc.) are inconsistent

Errors detected in the sending and receiving phases will cause TXERR or RXERR to be incremented by one. According to the range of TXERR and RXERR, there will be three error states. Different states will have different operations for new errors detected:

Active error state: send active error frame (6 dominant bits);
Passive error state: send passive error frame (6 recessive bits);
Bus off state: Stop all sending behaviors, entering reset mode, automatically pulling up RM, modifying TXERR=127. in these ways the next initialization needs to wait for 11 consecutive recessive bits for 128 times to end. Normal power-on only needs to wait 1 time;



20.5.7 Failure Arbitration Address Number

In the arbitration phase, the identifiers will be compared in turn. After the arbitration fails, the ALC can be read to find the specific location of the failure::

Value of ALC	Standard frame failure address	Expansion framefailure address
31	-	RTR bit
30	-	ID 0 bit
29	-	ID 1 bit
28	-	ID 2 bit
27	-	ID 3 bit
26	-	ID 4 bit
25	-	ID 5 bit
24	-	ID 6 bit
23	-	ID 7 bit
22	-	ID 8 bit
21	-	ID 9 bit
20	-	ID 10 bit
19	-	ID 11 bit
18	-	ID 12 bit
17	-	ID 13 bit
16	-	ID 14 bit
15	-	ID 15 bit
14	-	ID 16 bit
13	-	ID 17 bit
12	-	IDE bit
11	RTR bit	SRTR bit
10	ID 0 bit	ID 18 bit
9	ID 1 bit	ID 19 bit
8	ID 2 bit	ID 20 bit
7	ID 3 bit	ID 21 bit
6	ID 4 bit	ID 22 bit
5	ID 5 bit	ID 23 bit
4	ID 6 bit	ID 24 bit
3	ID 7 bit	ID 25 bit
2	ID 8 bit	ID 26 bit
1	ID 9 bit	ID 27 bit
0	ID 10 bit	ID 28 bit

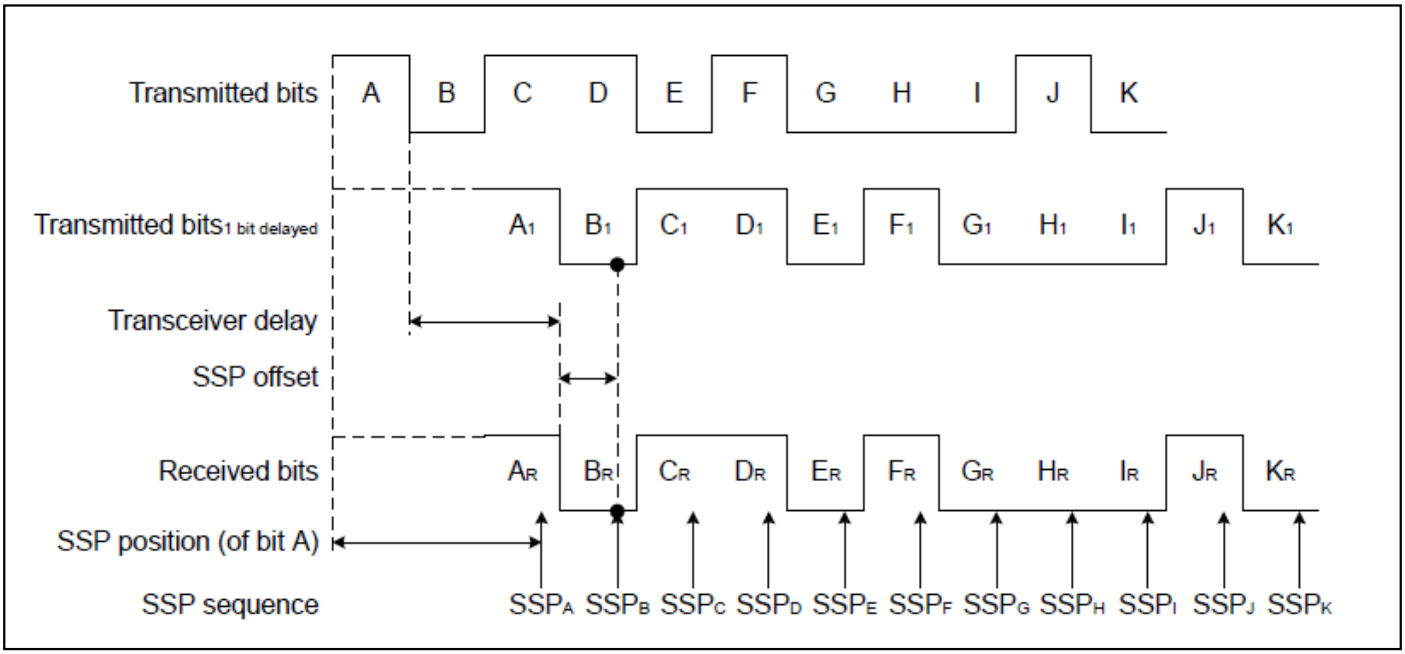
20.5.8 Transmit and Receive Delay Compensation and Resampling

When sending, the sending end needs to compare and check each sent value and received value to ensure that the sending is correct. In the actual circuit, there will be a certain delay from sending to receiving, and the delay is generally less than one bit width, so the impact is not great. However, the data segment of the FD frame may use a very high rate. At this time, the sending and receiving delay will have a great impact on the sending and receiving inspection

The solution is to increase sending and receiving delay compensation and resampling:

- During the arbitration phase, the FDF falling edge is used to detect the estimated value of the transmit and receive delay;
- After the data stage, an adjustment value is added to the estimated value for resampling;

- the resampling position will have a delay of the estimated value + adjustment value compared with the normal sampling position;
- the received value and the sent value of the resampled position is used as the examine;

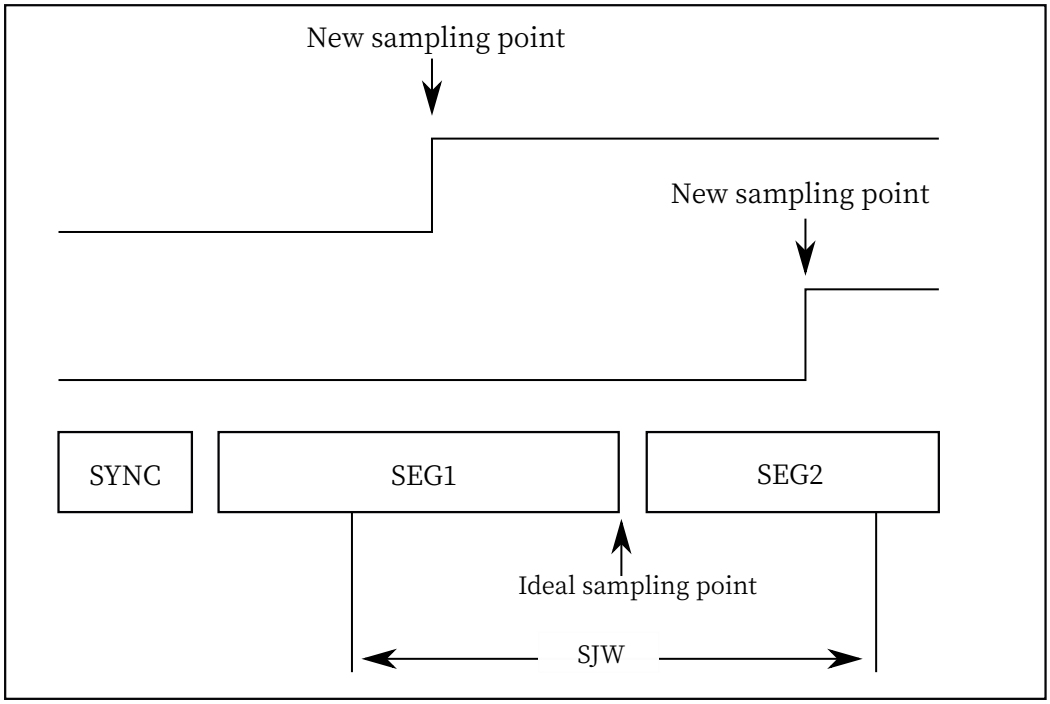


20.6 Bit Time Set

Each bit of CAN is composed of 3 parts, and each part contains a different number of time base units (TQ):

- Synchronization segment: Fixing 1 TQ;
- Time period 1: Configurable, between 1 ~16 TQ;
- Time period 2: Configurable, between 1 ~8 TQ;

The sampling point is generally at the end of time period 1 by default, and it is automatically adjusted with the synchronization mechanism: when the flip position is not at the end of time period 1, the sampling point after adjustment according to the flip position will move forward (before time period 1) or back (time period 2); in order to prevent the jump distance from being too large, a maximum jump distance is generally set: SJWs.



In order to improve the accuracy of sampling, the controller can use 3 sampling modes: the ideal sampling point and the previous 2 TQ sampling, and the result of the 3 is the final result. This mode works well in medium and low speed communication.

Double rate may be used in FD frames, so there will be different rate configurations. There are 3 bit time settings. Traditional frames only use BIT1/0 while FD frames can use NBT and DBT:

	BIT1/0	NBT	DBT
BRP	TBRP	NBRP	DBRP
SEG1	TSEG1	NSEG1	DSEG1
SEG2	TSEG2	NSEG2	DSEG2
SJW	TSJW	NSJW	DSJW

The FD frame can select the rate of each phase through BRSEN and EXTBN:

BRSEN	EXTBT	FD Frame arbitration phase	FD Frame data phase
0	0	BIT1/0	BIT1/0
0	1	NBT	NBT
1	0	BIT1/0	DBT
1	1	NBT	DBT

20.7 bxCAN Register

Tab 20.7-1 bxCAN register map

offset	Register	Reset value	Description
bxCAN = 0x4000_6400			
0x00	ISR_SR_CMR_MR	0x0020_0004	ISR_SR_CMR_MR
0x04	BT1_BT0_RMC_IMR	0x0000_0000	BT1_BT0_RMC_IMR
0x08	TXBUF	0x0000_0000	TXBUF
0x0C	RXBUF	0x100C_C34C	TXBUF
0x10	ACR	0x0000_0000	ACR
0x14	AMR	0x0000_0000	AMR
0x18	ALC_TXERR_RXERR_ECC	0x0000_0000	ALC_TXERR_RXERR_ECC
0x1C	NBT	0x0000_0000	NBT
0x20	SSPP_TDCR_DBT	0x0000_0000	SSPP_TDCR_DBT
0x24	APERR_DPERR_FDSTA_FDCFG	0x0000_0000	APERR_DPERR_FDSTA_FDCFG
0x28	TEST	0x0000_0000	TEST

Note: *In order to prevent abnormalities caused by modifying parameters during the working process of the controller, some registers will be locked when exiting the reset mode. Such lockable registers are marked with "Lock" in the Remark.*

20.7.1 ISR_SR_CMR_MR

Register	Address offset	Access	Reset value
ISR_SR_CMR_MR	0x00	RW	0x0020_0004

31	30	29	28	27	26	25	24
Reserved	ALI	EWI	EPI	RI	TI	BEI	DOI
23	22	21	20	19	18	17	16
RBS	DSO	TBS	Reserved	RS	TS	ES	BS
15	14	13	12	11	10	9	8
Reserved					TR	AT	Reserved
7	6	5	4	3	2	1	0
Reserved					RM	LOM	AFM

ISR_SR_CMR_MR bit description

Bit	Access	Description
[31]	R	Reserved
[30]	RW	**ALI:** Failed arbitration interruption, 1: Arbitration has failed and has enteredreceive mode, at this time the ALC registercan be read to see which frame has failedat this time 1 can be cleared; 0: Successful arbitration;
[29]	RW	**EWI:** Error warning interrupt 1: the 2 bits ES(bit17)/BS (bit16) have changed; at thistime, write 1 can be cleared: 0: no change in ES/BS has occurred; **Note:** *Note: In fact, it can be simply understood thathe interrupt is generated when the errol-counter is not in the range [96,255], since EScan be simply equated to a counter lessthan 96 and BS equated to greater than 255*

[28]	RW	**EPI:** Passive error interruptions 1: has entered or exited a passive error state, at this time, wriing 1 can can be cleared 0: no entry or exit from the passive errorstate has occurred:
[27]	RW	**RI:** Receiving interruptions 1: At least one reception is successful; at this time, write 1 can reduce the RMC minus one, so each time after reading a set of messages from RX MEM After reading a set of messages from RX MEM, the bit should be written once to 1 0: At this point the reception is unsuccessful or incomplete
[26]	RW	**TI:** Transmission interruption 1: This send was successful; at this point write 1 can be cleared and the reset reset the pointer to the transmit cache, so that before the next transmit, the bit should be written once before the next transmission 0: the transmission was unsuccessful or incomplete;
[25]	RW	**BEI:** Bus error interrupts; 1: an error has occurred during reception and transmission, when write 1 can be cleared; 0: no bus error has occurred;
[24]	RW	**DOI:** Data overflow interrupt 1: RX_MEM has had an overflow event, at which point a write 1 can clear zero; 0: no overflow event has occurred in RX_MEM;
[23]	RW	**RBS:** Receive buffer status; 1: at least one message has been deposited in the RX_MEM; 0: RX_MEM is empty;
[22]	R	**DSO:** Data overflow status 1: RX_MEM has overflowed; 0: RX_MEM is normal;
[21]	R	**TBS:** The status of the transmit buffer 1: CPU writable; 0: locked state, CPU unwritable
[20]	R	Reserved
[19]	R	**RS:** Receiving status; 1: a message is being received; 0: no messages are being received;
[18]	R	**TS:** Transmission status 1: messages are being sent; 0: no message is being sent
[17]	R	**ES:** Error states 1: one of the receive and transmit error counters reaches 96; 0: both error counters are less than 96
[16]	R	**BS:** Bus status 1: bus off, no bus connection available; 0: bus is on, transceiver operation is possible; According to CAN protocol, when exiting from reset mode, it will work in the active error state by default. When the error counter exceeds 127, it will enter the passive error state. If the error counter continues to increase beyond 255, it will enter the bus closed state to prevent nodes from causing too much interference to the bus
[15:11]	R	Reserved

[10]	W	**TR:** transfer request, only 1 can be written Start a message sending process after writing; if you want to terminate this sending, you need to write 1 to AT
[9]	W	**AT:** Terminate the transmission request; only 1 can be written; Write 1 to AT at the same time as TR, this message sending process is executed only once, including error occurrence, arbitration failure, loss of The retransmission process will not be started due to various conditions, including error occurrence, arbitration failure, loss of frames, etc. The retransmission process is not started; Writing 1 to the AT after writing 1 to the TR may stop the sending process. The message sending process may be stopped by writing 1 to AT after TR writes 1, depending on whether the sending process If it is already started, it cannot be terminated.
[8:3]	R	Reserved
[2]	RM	**RM:** Reset mode switch 0: working mode, then select 2 working modes according to LOM working mode; 1: reset mode, all functions are turned off, no receiving and no sending. Most registers are writable only in this mode;
[1]	RM	**LOM:** silent mode switch, writable in reset mode only; 0: silent mode off, can receive and send; 1: open silent mode, only receive but not send **Note:** *Note: In silent mode, even the received ACKs are not returned. and the error counters are not updated. This mode is mainly used to detect the bit rate or to do other CAN bus Data Analysis*
[0]	RM	**AFM:** Receive filter mode selection, writable in reset mode only; 0: use of dual filter; 1: using a single filter;

20.7.2 BT1_BT0_RMC_IMR

Register	Address offset	Access	Reset value
BT1_BT0_RMC_IMR	0x04	RW	0x0000_0000

31	30	29	28	27	26	25	24
SAM	TSEG2			TSEG1			
23	22	21	20	19	18	17	16
TSJW		TBRP					
15	14	13	12	11	10	9	8
Reserved			RMC				
7	6	5	4	3	2	1	0
Reserved	ALIM	EWIM	EPIM	RIM	TIM	BEIM	DOIM

BT1_BT0_RMC_IMR bit description

Bit	Access	Description
[31]	RW	**SAM:** Sampling mode 1:three sampling selections the sampling position is the last TQ of TSEG1 and the previous two recommended for low and mediun sped modes 0: single sampling, samplingoosition for TSEG1 most **Note:** *The latter TQ, recommended for use in high-speed mode*
[30: 28]	RW	**TSEG2:** Length of time period 1 = TQ*(TSEG2+1)

[27:24]	RW	**TSEG1:** Length of time period 1 = TQ*(TSEG1+1)
[23:22]	RW	**TSJW:** Maximum synchronous jump width; 0: 1 TQ; 1: 2 TQs; 2: 3 TQs; 3: 4 TQs;
[21:16]	RW	**TBRP:** CAN base unit TQ prescale multiplier the system clock of the CAN controller is PCLK1so the base unit frequency TQ=PCLK1*2*(TBRP+1)
[15:14]	R	Reserved
[12:8]	R	**RMC:** Received message counter, for each successfully received message Add one after the text and write 1minus one each time to the ISR
[7]	R	Reserved
[6]	RW	**ALIM:** ALI interrupt switch; 0: off; 1: on;
[5]	RW	**EWIM:** EWl interrupt switch; 0: off; 1: on
[4]	RW	**EPIM:** EPI interrupt switch; 0: off, 1: on;
[3]	RW	**RIM:** Rl interrupt switch; 0: off; 1: on;
[2]	RW	**TIM:** TI interrupt switch; 0: off; 1: on;
[1]	RW	**BEIM:** BEI interrupt switch; 0: off; 1: on;
[0]	RW	**DOIM:** DOI interrupt switch; 0: off; 1: on;

20.7.3 TXBUF

Register	Address offset	Access	Reset value
TXBUF	0x08	R	0x0000_0000

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

TXBUF bit description

Bit	Access	Description
[31:0]	W	**TXBUF:** Send cache configuration interface, each write value will be filled by hardware TX_MEM, the MEM address is automatically added by one, and the address is cleared when giving TI is cleared on a write of one;

20.7.4 RXBUF

Register	Address offset	Access	Reset value
RXBUF	0x0C	R	0x100C_C34C

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

TXBUF bit description

Bit	Access	Description
[31:0]	W	**RXBUF:** Receive cache matching interface, each read value will be taken from RX_MEM by the hardware once, MEM address automatically added one, the address can not be cleared directly;

20.7.5 ACR

Register	Address offset	Access	Reset value
ACR	0x10	R	0x0000_0000

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

ACR description

Bit	Access	Description
[31:0]	RW	**ACR:** Acceptance filter match value register

20.7.6 AMR

Register	Address offset	Access	Reset value
AMR	0x14	R	0x0000_0000

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

ACR description

Bit	Access	Description
[31:0]	RW	**AMR:** Acceptance filter mask value register, each corresponding to a check bit; set 1 means no need to check, set 0 means need to compare with ACR; all bits match means acceptance pass All of the bits

20.7.7 ALC_TXERR_RXERR_ECC

Register	Address offset	Access	Reset value
ALC_TXERR_RXERR_ECC	0x18	R	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			ALC				
23	22	21	20	19	18	17	16
TXERR							
15	14	13	12	11	10	9	8
RXERR							
7	6	5	4	3	2	1	0
RXWRN	TXWRN	EDIR	ACKER	FRMER	CRCER	STFER	BER

ALC_TXERR_RXERR_ECC bit description

Bit	Access	Description
[31:29]	R	Reserved
[28:24]	R	**ALC:** Arbitration failure position number register
[23:16]	R	**TXERR:** Send error counter, add one for each error found during the sending process error counter; note that the actual counter itself has 9 but it can only read 8 bits; in addition, every time it enters the bus the counter automatically becomes 127 each time the line is closed;
[15:8]	R	**RXERR:** Receive error counter, when one error is not found in the receiving process When the bus is turned off, the counter is automatically cleared to zero. The counter is automatically cleared when the bus is turned off.
[7]	R	**RXWRN:** When the RXERR counter is greater than or equal to 96 Pull High
[6]	R	**TXWRN:** When the RXERR counter is greater than or equal to 96 Pull High
[5]	R	**EDIR:** The direction of transmission in case of an error. 0: Send 1: Receiving
[4]	R	**ACKER:** ACK error occurred: ACK SLOT is a hidden bit
[3]	R	**FRMER:** A format error occurred: The value of thefixed dominant/implicit bit is not Right
[2]	R	**CRCER:** CRC error occurred: CRC sequence check sum error
[1]	R	**STFER:** Filling error: Consecutive identica values with more than 5 bits
[0]	R	**BER:** A bit error occurred: the sent value andthe listened value are different

20.7.8 NBT

Note: *NBT only takes effect when EXTBT is 1. When EXTBT is 0, it is still configured with TBRP etc. in BT1/BT0.*

Register	Address offset	Access	Reset value
NBT	0x1C	RW	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
NSJM				NSEG2			
15	14	13	12	11	10	9	8
NSEG1						NBRP	
7	6	5	4	3	2	1	0
NBRP							

NBT bit description

Bit	Access	Description
[31:24]	R	Reserved
[23:20]	RW	**NSJM:** Maximum synchronous hopping distance of the arbitration phase at FD frame: 0 ~ 15: 1 ~ 16 NTQs;
[19:16]	RW	**NSEG2:** Length of time after the sampling point ofthe arbitration phase at FD frame. NTQ*(NSEG2+1)
[15:10]	RW	**NSEG1:** Length of time before the sampling point ofthe arbitration phase at FD frame: NTQ*(NSEG1+1)
[9:0]	RW	**NBRP:** Frequency of NTQ in arbitration phase at FD frame: PCLK1*2*(NBRP+1)

20.7.9 SSPP_TDCR_DBT

Note: *DBT is effective only when FDEN is 1, and when FDEN is 0 the data phase is maintained using the configuration of NBT, etc.*

Register	Address offset	Access	Reset value
SSPP_TDCR_DBT	0x20	RW	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	SSPP						
23	22	21	20	19	18	17	16
TDCEN	TDCO						
15	14	13	12	11	10	9	8
DSEG2			DSEG1				
7	6	5	4	3	2	1	0
DSJW			DBRP				

SSPP_TDCR_DBT bit description

Bit	Access	Description
[31]	R	Reserved
[30:24]	RW	**SSPP:** The final location of the secondary sampling point, containing the estimated value and adjusted values
[23]	RW	**TDCEN:** Transceiver delay compensation enable. 1: On 0: off
[22:16]	RW	**TDCO:** Transceiver delay compensation adjustment value
[15:13]	RW	**DSEG2:** Length of time after the sampling point of the data phase at FD frame: DTQ*（DSEG2+1）
[12:8]	RW	**DSEG1:** Length of time after the sampling point of the data phase at FD frame: DTQ*（DSEG1+1）
[7:5]	RW	**DSJW:** Maximum synchronous hopping distance of data phase at FD frame: 0 ~ 15: 1 ~ 16 DTQs;
[4:0]	RW	**DBRP:** Frequency of DTQ in data phase at FD frame: PCLK1*2*(DBRP+1)

20.7.10 APERR_DPERR_FDSTA_FDCFG

Register	Address offset	Access	Reset value
APERR_DPERR_FDSTA_FDCFG	0x24	R	0x0000_0000

31	30	29	28	27	26	25	24
APERR							
23	22	21	20	19	18	17	16
DPXERR							
15	14	13	12	11	10	9	8
STATE		Reserved		STFERR	FRMERR	CRCERR	BITERR
7	6	5	4	3	2	1	0
Reserved		REOM	DAR	ISO	EXTBT	BRSEN	FDEN

APERR_DPERR_FDSTA_FDCFG bit description

Bit	Access	Description
[31:24]	R	**APERR:** Arbitration segment error counter, added one each time an error occurs in the arbitration phase error occurs; max. 255
[23:16]	R	**DPXERR:** Data segment error counter, each time a frame is sent or received in the data phase error occurs, plus one; maximum 255;
[15:14]	R	**STATE:** Controller operation status: 00: Initial state, each reset or bus off after each reset or bus shutdown, until the end of 11 consecutive implicit bits are received The controller operates in the following states 01: idle state, waiting for the frame header 10: Receive state, receiving; 11: transmit state, transmitting;
[13:12]	R	Reserved
[11]	R	**STFERR:** When BRS bit is 1, it works. 1 indicates that a bit filling error occurred in the FD frame data phase
[10]	R	**FRMERR:** BRS bit works only when it is 1. A 1 indicates that an FD frame data phase an illegal bit error has occurred;
[9]	R	**CRCERR:** A value of 1 indicates a CRC checksum error in the FD frame;
[8]	R	**BITERR:** A value of 1 indicates that an error was found in the send/receive check of the FD frame
[7:6]	R	Reserved
[5]	RW	**REOM:** Restricted operation modes: 1: Restricted operation mode is on and the node does not send error frames (active or passive) or overload frames. When the error error or overload condition is met, no explicit bit is sent, but the node directly node directly enters the wait bus idle state; 0: normal mode, error frames (active or passive) are sent when the corresponding conditions are met.
[4]	RW	**DAR:** Disabling automatic retransmission; 1: Disable automatic retransmission 0: Enable automatic retransmission
[3]	RW	**ISO:** Format options; 1: ISO 11898-1:2015 format 0: Bosch format
[2]	RW	**EXTBT:** This bit configures the bit-time prescaler for the arbitration phase 1: Use NBTL, NBTH and NBTU registers during the arbitration phase using the contents of the NBTL, NBTH and NBTU registers to configure the bit time 0: Use the BT0 and BT1 registers in the arbitration phase Content configuration bit time
[1]	RW	**BRSEN:** This bit indicates whether to switch the bit rate during the data phase 1: switches to a bit rate dedicated to the data phase (DBT configuration); 0: maintains the bit rate of the arbitration phase;

[0]	RW	**FDEN:** Frame format 1: FD frame format 0: Traditional frame format

20.7.11 TEST

Register	Address offset	Access	Reset value
TEST	0x28	RW	0x0000_0000

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

TEST bit description

Bit	Access	Description
[31:2]	R	Reserved
[1]	RW	**TXC:** Send signal control switch when LBEN is on; 0: The signal is sent to the receiving end and also sent to the external bus; 1: The sending signal is not sent to the external bus, but only to the receiving end. At this time, the sending end will be fixed into a hidden bit;
[0]	RW	**LBEN:** The self-test loop is enabled, and when it is turned on, it can directly send the transmit transmit signal from internal to the receiver, at this time the receiver end no longer receives signals coming in from the external bus; 0: off 1: open

21 Serial peripheral interface (SPI)

The SPI interface provides two main functions, supporting either the SPI protocol or the I2S audio protocol. By default, it is the SPI function that is selected. It is possible to switch the interface from SPI to I2S by software.

The serial peripheral interface (SPI) allows half/ full-duplex, synchronous, serial communication with external devices. The interface can be configured as the master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

It may be used for a variety of purposes, including simplex synchronous transfers on two lines with a possible bidirectional data line or reliable communication using CRC checking. The I2S is also a synchronous serial communication interface. It can address four different audio standards including the I2S Philips standard, the MSB- and LSB-justified standards, and the PCM standard. It can operate as a slave or a master device in full-duplex mode (using 4 pins) or in half-duplex mode (using 6 pins). Master clock can be provided by the interface to an external slave component when the I2S is configured as the communication master.

21.1 SPI and I2S main features

21.1.1 SPI features

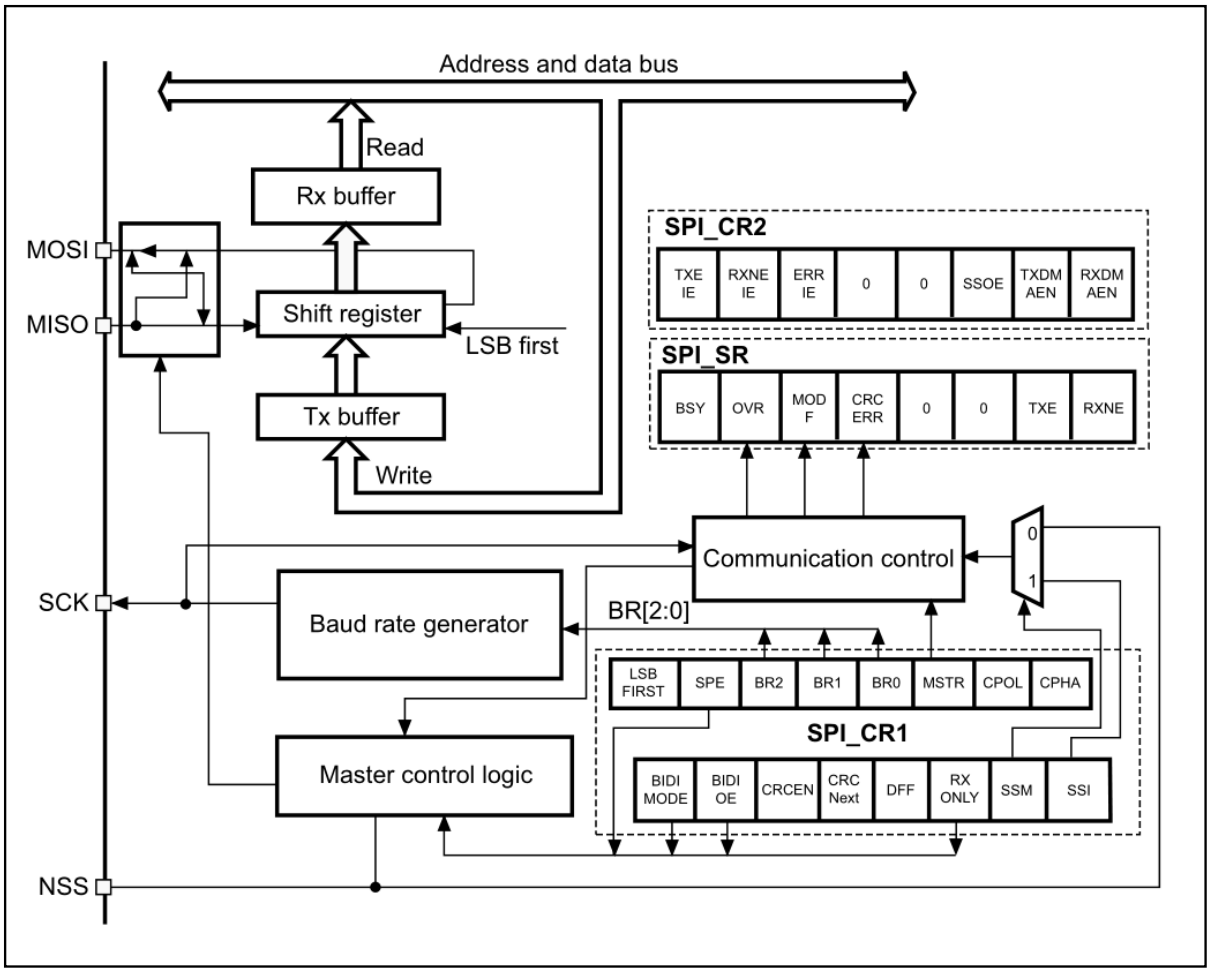
- Full-duplex synchronous transfers on three lines
- Simplex synchronous transfers on two lines with or without a bidirectional data line
- 8- or 16-bit transfer frame format selection
- Master or slave operation
- Multimaster mode capability
- 8 master mode baud rate prescalers (f PCLK /2 max.)
- Slave mode frequency (f PCLK /2 max)
- Faster communication for both master and slave
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun and CRC error flags with interrupt capability
- 1-byte transmission and reception buffer with DMA capability: Tx and Rx requests

21.1.2 I2S features

- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode and Overrun flag in reception mode (master and slave)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I2S protocols:
  - I2S Phillps standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock may be output to drive an external audio component. Ratio is fixed at 256 × F_S (where F_S is the audio sampling frequency)
- In connectivity line devices, both I2S (I2S2 and I2S3) have a dedicated PLL (PLL3) to generate an even more accurate clock.

21.2 SPI General description

Fig 21.2-1 SPI block diagram



21.2.1 SPI agreement

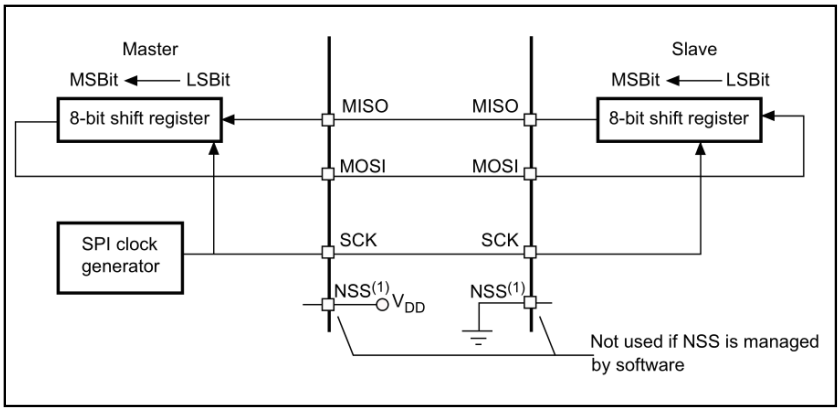
Typically the SPI is connected to external devices by four pins

- MISO: Master In / Slave Out data. This pin can be used to transmit data in slave mode and receive data in master mode.
- MOSI: Master Out / Slave In data. This pin can be used to transmit data in master mode and receive data in slave mode.
- SCK: Serial Clock output for SPI masters and input for SPI slaves.
- NSS: Slave select. This is an optional pin to select a slave device. This pin acts as a ‘chip select’ to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave NSS inputs can be driven by standard IO ports on the master device. The NSS pin may also be used as an output if enabled (SSOE bit) and driven low if the SPI is in master configuration. In this manner, all NSS pins from devices connected to the Master NSS pin see a low level and become slaves when they are configured in NSS hardware mode. When configured in master mode with NSS configured as an input (MSTR=1 and SSOE=0) and if NSS is pulled low, the SPI enters the master mode fault state: the MSTR bit is automatically cleared and the device is

configured in slave mode (refer to Section 21.2.10).

A basic example of interconnections between a single master and a single slave is illustrated in Figure 21.2-2.

Fig 21.2-2 Single master/ single slave application



Note: *Here, the NSS pin is configured as an input.*

The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first). The communication is always initiated by the master. When the master device transmits data to a slave device via the MOSI pin, the slave device responds via the MISO pin. This implies full-duplex communication with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

Slave select (NSS) pin management

Hardware or software slave select management can be set using the SSM bit in the SPI_CR1 register.

- Software NSS management (SSM = 1)
  The slave select information is driven internally by the value of the SSI bit in the SPI_CR1 register. The external NSS pin remains free for other application uses.
- Hardware NSS management (SSM = 0)
  Two configurations are possible depending on the NSS output configuration (SSOE bit in register SPI_CR2).
  - NSS output enabled (SSM = 0, SSOE = 1)
    This configuration is used only when the device operates in master mode. The NSS signal is driven low when the master starts the communication and is kept low until the SPI is disabled.
  - NSS output disabled (SSM = 0, SSOE = 0)
    This configuration allows multimaster capability for devices operating in master mode. For devices set as slave, the NSS pin acts as a classical NSS input: the slave is selected when NSS is low and deselected when NSS high.

Clock phase and clock polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CR1 register. The CPOL (clock polarity) bit controls the steady state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA (clock phase) bit is set, the second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data are latched on the occurrence of the second clock transition. If the CPHA bit is reset, the first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data are latched on the occurrence of the first clock transition.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

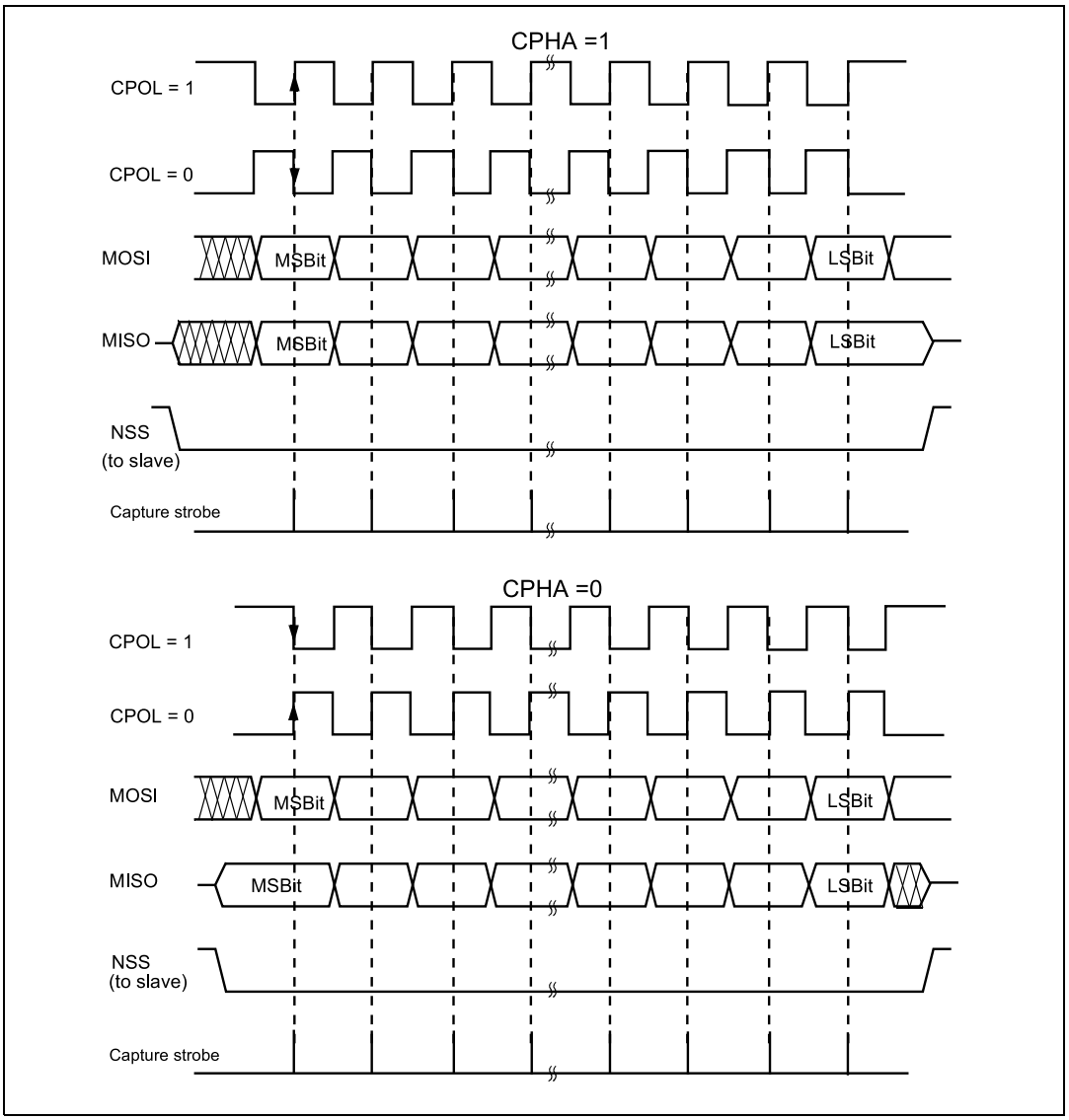
Figure 240, shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

Note: *Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit. Master and slave must be programmed with the same timing mode.*

The idle state of F_SCK must correspond to the polarity selected in the SPI_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

The Data Frame Format (8- or 16-bit) is selected through the DFF bit in SPI_CR1 register, and determines the data length during transmission/reception.

Fig 21.2-3 Data clock timing diagram



1. These timings are shown with the LSBFIRST bit reset in the SPI_CR1 register.

Data frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 Register.

Each data frame is 8 or 16 bits long depending on the size of the data programmed using the DFF bit in the SPI_CR1 register. The selected data frame format is applicable for transmission and/or reception.

21.2.2 Configuring the SPI in slave mode

In the slave configuration, the serial clock is received on the SCK pin from the master device. The value set in the BR[2:0] bits in the SPI_CR1 register, does not affect the data transfer rate.

Note: *It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave needs to be ready before the first edge of the communication clock or before the*

end of the ongoing communication. It is mandatory to have the polarity of the communication clock set to the steady state value before the slave and the master are enabled.

Follow the procedure below to configure the SPI in slave mode:

Procedure

1. Set the DFF bit to define 8- or 16-bit data frame format
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see Figure 21.2-3). For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and the master device.
3. The frame format (MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 register) must be the same as the master device.
4. In Hardware mode (refer to Section：21.2.1 Slave select (NSS) pin management), the NSS pin must be connected to a low level signal during the complete byte transmit sequence. In NSS software mode, set the SSM bit and clear the SSI bit in the SPI_CR1 register.
5. Clear the MSTR bit and set the SPE bit (both in the SPI_CR1 register) to assign the pins to alternate functions.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

Transmit sequence

The data byte is parallel-loaded into the Tx buffer during a write cycle.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin. The remaining bits (the 7 bits in 8-bit data frame format, and the 15 bits in 16-bit data frame format) are loaded into the shift-register. The TXE flag in the SPI_SR register is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The Data in shift register is transferred to Rx Buffer and the RXNE flag (SPI_SR register) is set
- An Interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register.

After the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.

Clearing of the RXNE bit is performed by reading the SPI_DR register.

21.2.3 Configuring the SPI in master mode

In the master configuration, the serial clock is generated on the SCK pin.

Procedure

1. Select the BR[2:0] bits to define the serial clock baud rate (see SPI_CR1 register).
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see Figure 21.2-3).
3. Set the DFF bit to define 8- or 16-bit data frame format
4. Configure the LSBFIRST bit in the SPI_CR1 register to define the frame format.
5. If the NSS pin is required in input mode, in hardware mode, connect the NSS pin to a high-level signal during the complete byte transmit sequence. In NSS software mode, set the SSM and SSI bits in the SPI_CR1 register. If the NSS pin is required in output mode, the SSOE bit only should be set.
6. The MSTR and SPE bits must be set (they remain set only if the NSS pin is connected to a high-level signal).

In this configuration the MOSI pin is a data output and the MISO pin is a data input.

Transmit sequence

The transmit sequence begins when a byte is written in the Tx Buffer. The data byte is parallel-loaded into the shift register (from the internal bus) during the first bit transmission and then shifted out serially to the MOSI pin MSB first or LSB first depending on the LSBFIRST bit in the SPI_CR1 register. The TXE flag is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The data in the shift register is transferred to the RX Buffer and the RXNE flag is set
- An interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register

At the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.

Clearing the RXNE bit is performed by reading the SPI_DR register.

A continuous transmit stream can be maintained if the next data to be transmitted is put in the Tx buffer once the transmission is started. Note that TXE flag should be ‘1 before any attempt to write the Tx buffer is made.

Note: *When a master is communicating with SPI slaves which need to be de-selected between transmissions, the NSS pin must be configured as GPIO or another GPIO must be used and toggled by software*

21.2.4 Configuring the SPI for half-duplex communication

The SPI is capable of operating in half-duplex mode in 2 configurations.

- 1 clock and 1 bidirectional data wire
- 1 clock and 1 data wire (receive-only or transmit-only)

1 clock and 1 bidirectional data wire (BIDIMODE = 1)

This mode is enabled by setting the BIDIMODE bit in the SPI_CR1 register. In this mode SCK is used for the clock and MOSI in master or MISO in slave mode is used for data communication. The transfer direction (Input/Output) is selected by the BIDIOE bit in the SPI_CR1 register. When this bit is 1, the data line is output otherwise it is input.

1 clock and 1 unidirectional data wire (BIDIMODE = 0)

In this mode, the application can use the SPI either in transmit-only mode or in receive-only mode.

- Transmit-only mode is similar to full-duplex mode (BIDIMODE=0, RXONLY=0): the data are transmitted on the transmit pin (MOSI in master mode or MISO in slave mode) and the receive pin (MISO in master mode or MOSI in slave mode) can be used as a general-purpose IO. In this case, the application just needs to ignore the Rx buffer (if the data register is read, it does not contain the received value).
- In receive-only mode, the application can disable the SPI output function by setting the RXONLY bit in the SPI_CR1 register. In this case, it frees the transmit IO pin (MOSI in master mode or MISO in slave mode), so it can be used for other purposes

To start the communication in receive-only mode, configure and enable the SPI:

- In master mode, the communication starts immediately and stops when the SPE bit is cleared and the current reception stops. There is no need to read the BSY flag in this mode. It is always set when an SPI communication is ongoing.
- In slave mode, the SPI continues to receive as long as the NSS is pulled down (or the SSI bit is cleared in NSS software mode) and the SCK is running.

21.2.5 Data transmission and reception procedures

Rx and Tx buffers

In reception, data are received and then stored into an internal Rx buffer while In transmission, data are first stored into an internal Tx buffer before being transmitted.

A read access of the SPI_DR register returns the Rx buffered value whereas a write access to the SPI_DR stores the written data into the Tx buffer.

Start sequence in master mode

- In full-duplex (BIDIMODE=0 and RXONLY=0)
  - The sequence begins when data are written into the SPI_DR register (Tx buffer).
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
  - At the same time, the received data on the MISO pin is shifted in serially to the 8- bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
  - The sequence begins as soon as SPE=1

  - Only the receiver is activated and the received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
  - The sequence begins when data are written into the SPI_DR register (Tx buffer).
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
  - No data are received.
- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)
  - The sequence begins as soon as SPE=1 and BIDIOE=0.
  - The received data on the MOSI pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
  - The transmitter is not activated and no data are shifted out serially to the MOSI pin.

Start sequence in slave mode

- In full-duplex mode (BIDIMODE=0 and RXONLY=0)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
  - At the same time, the data are parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission, and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
  - The transmitter is not activated and no data are shifted out serially to the MISO pin.
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
  - The sequence begins when the slave device receives the clock signal and the first bit in the Tx buffer is transmitted on the MISO pin.
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
  - No data are received.
- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MISO pin.
  - The received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
  - The transmitter is not activated and no data are shifted out serially to the MISO pin.

Handling data transmission and reception

The TXE flag (Tx buffer empty) is set when the data are transferred from the Tx buffer to the shift register. It indicates that the internal Tx buffer is ready to be loaded with the next data. An interrupt can be generated if the TXEIE bit in the SPI_CR2 register is set. Clearing the TXE bit is performed by

writing to the SPI_DR register.

Note: *The software must ensure that the TXE flag is set to 1 before attempting to write to the Tx buffer. Otherwise, it overwrites the data previously written to the Tx buffer.*

The RXNE flag (Rx buffer not empty) is set on the last sampling clock edge, when the data are transferred from the shift register to the Rx buffer. It indicates that data are ready to be read from the SPI_DR register. An interrupt can be generated if the RXNEIE bit in the SPI_CR2 register is set. Clearing the RXNE bit is performed by reading the SPI_DR register.

For some configurations, the BSY flag can be used during the last data transfer to wait until the completion of the transfer.

Full-duplex transmit and receive procedure in master or slave mode (BIDIMODE=0 and RXONLY=0)

The software has to follow this procedure to transmit and receive data (see Figure 21.2-4 and Figure 21.2-5):

1. Enable the SPI by setting the SPE bit to 1.
2. Write the first data item to be transmitted into the SPI_DR register (this clears the TXE flag).
3. Wait until TXE=1 and write the second data item to be transmitted. Then wait until RXNE=1 and read the SPI_DR to get the first received data item (this clears the RXNE bit). Repeat this operation for each data item to be transmitted/received until the n–1 received data.
4. Wait until RXNE=1 and read the last received data.
5. Wait until TXE=1 and then wait until BSY=0 before disabling the SPI.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edges of the RXNE or TXE flag.

Fig 21.2-4 TXE/RXNE/BSY behavior in Master / full-duplex mode (BIDIMODE=0 and RXONLY=0) in case of continuous transfers

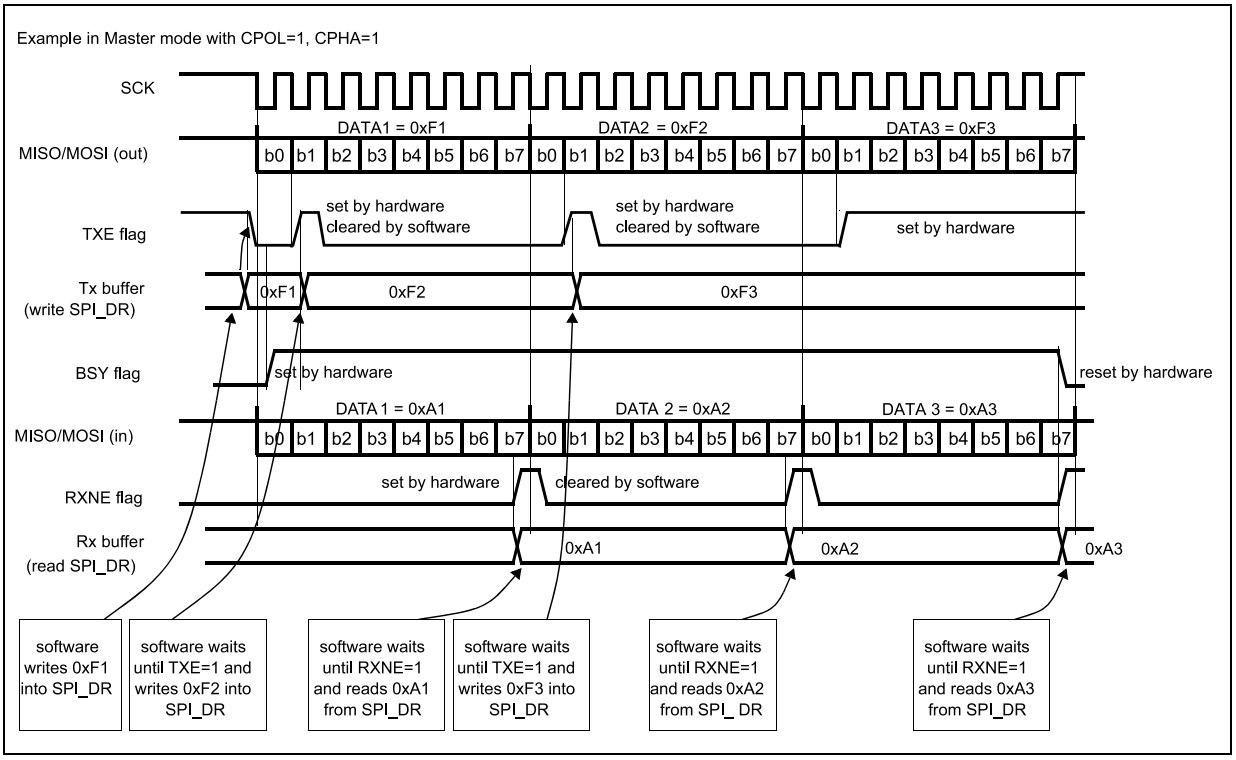
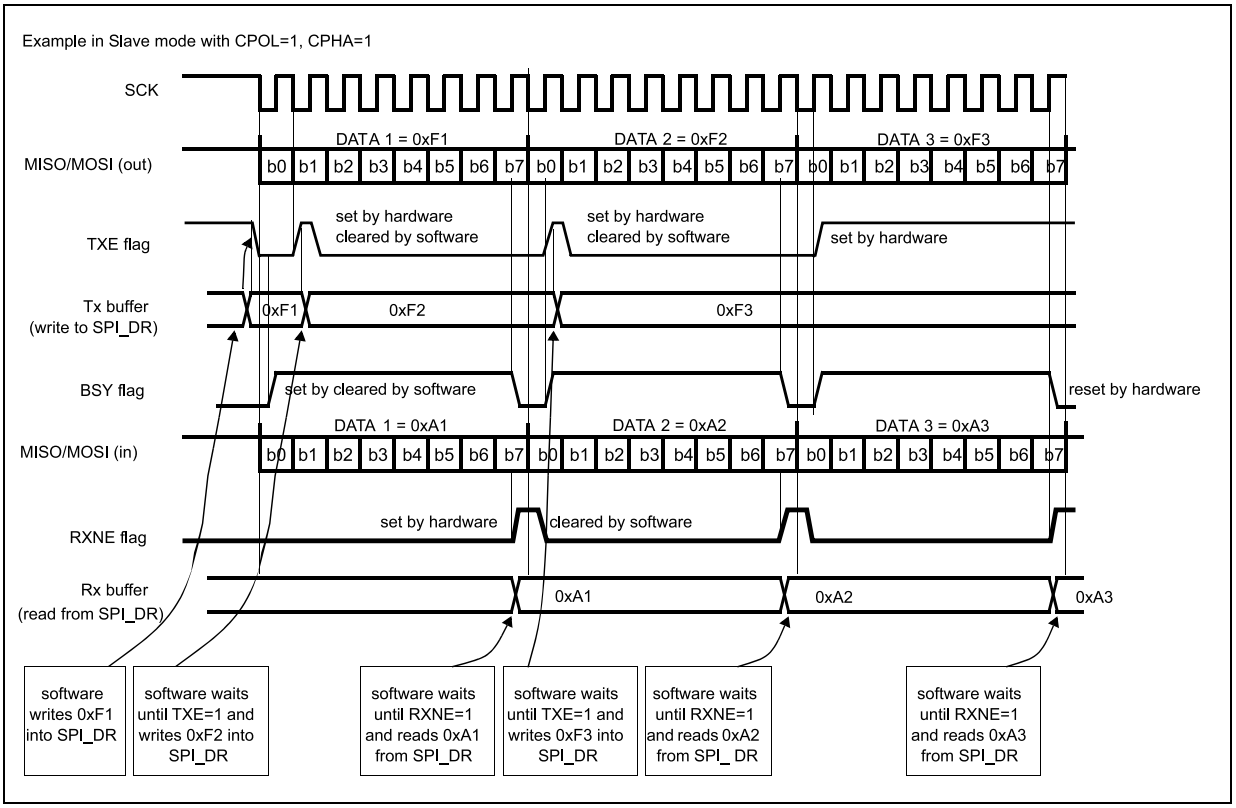


Fig 21.2-5 TXE/RXNE/BSY behavior in Slave / full-duplex mode (BIDIMODE=0, RXONLY=0) in case of continuous transfers



Transmit-only procedure (BIDIMODE=0 RXONLY=0)

In this mode, the procedure can be reduced as described below and the BSY bit can be used to wait until the completion of the transmission (see Figure 243 and Figure 244).

1. Enable the SPI by setting the SPE bit to 1.
2. Write the first data item to send into the SPI_DR register (this clears the TXE bit).
3. Wait until TXE=1 and write the next data item to be transmitted. Repeat this step for each data item to be transmitted.
4. After writing the last data item into the SPI_DR register, wait until TXE=1, then wait until BSY=0, this indicates that the transmission of the last data is complete.

This procedure can be also implemented using dedicated interrupt subroutines launched at each rising edge of the TXE flag.

Note: *During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, in transmit-only mode, it is mandatory to wait first until TXE is set and then until BSY is cleared after writing the last data.*

After transmitting two data items in transmit-only mode, the OVR flag is set in the SPI_SR register since the received data are never read.

Fig 21.2-6 TXE/BSY behavior in Master transmit-only mode (BIDIMODE=0 and RXONLY=0) in case of continuous transfers

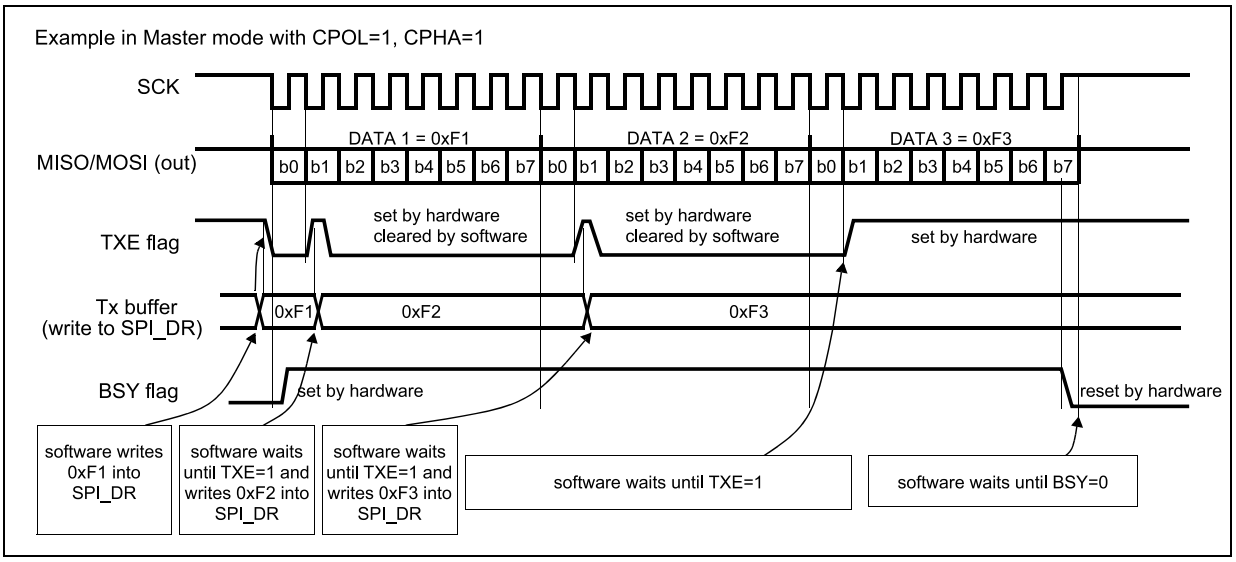
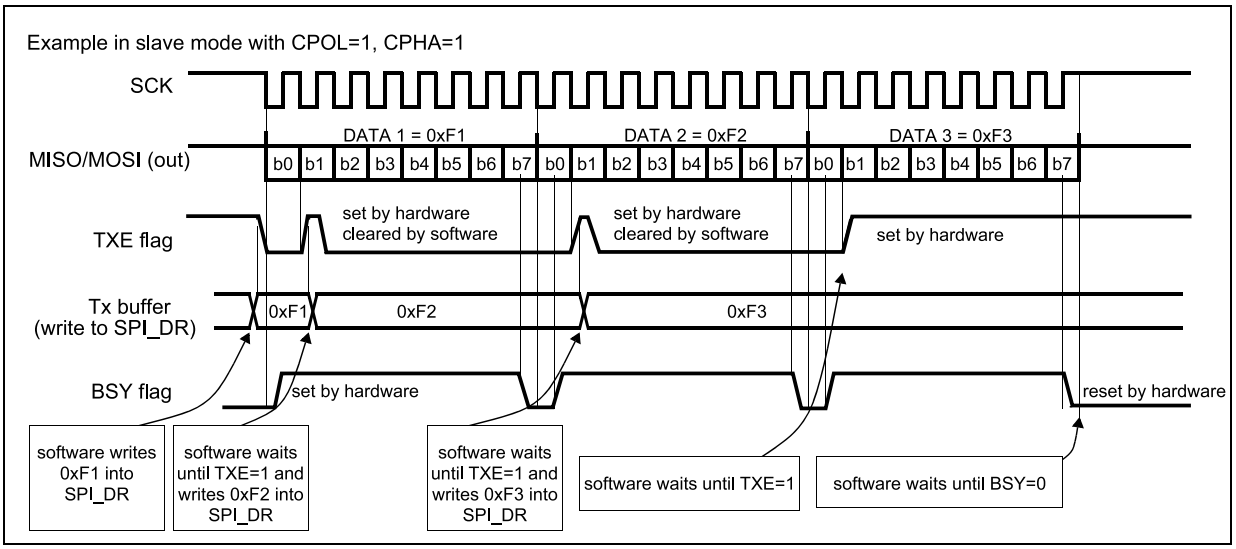


Fig 21.2-7 TXE/BSY in Slave transmit-only mode (BIDIMODE=0 and RXONLY=0) in case of continuous transfers



Bidirectional transmit procedure (BIDIMODE=1 and BIDIOE=1)

In this mode, the procedure is similar to the procedure in Transmit-only mode except that the BIDIMODE and BIDIOE bits both have to be set in the SPI_CR2 register before enabling the SPI.

Unidirectional receive-only procedure (BIDIMODE=0 and RXONLY=1)

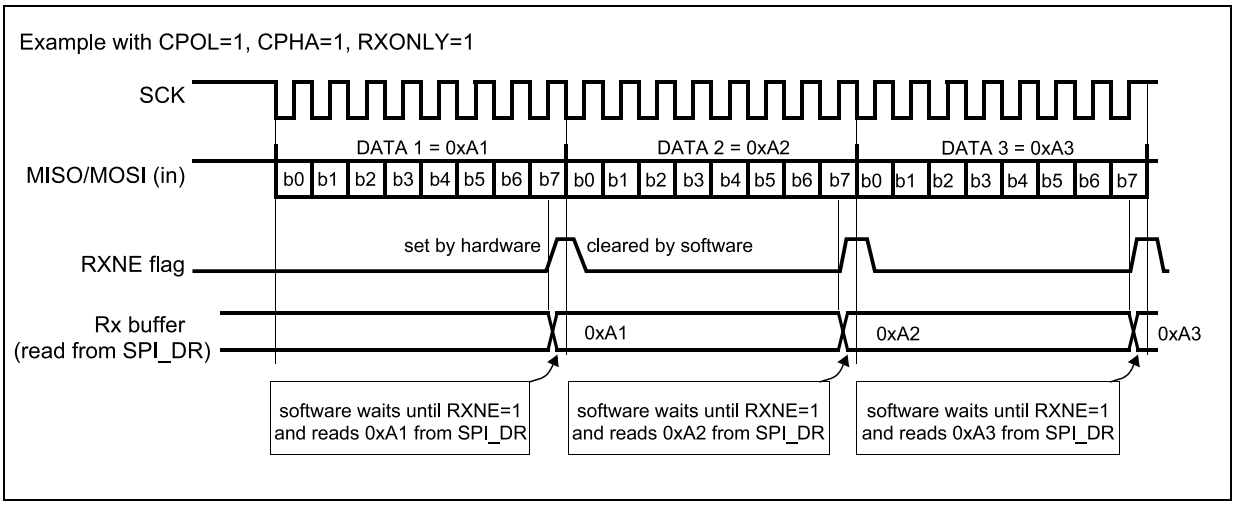
In this mode, the procedure can be reduced as described below (see Figure 21.2-8):

1. Set the RXONLY bit in the SPI_CR1 register.
2. Enable the SPI by setting the SPE bit to 1:
   - In master mode, this immediately activates the generation of the SCK clock, and data are serially received until the SPI is disabled (SPE=0).
   - In slave mode, data are received when the SPI master device drives NSS low and generates the SCK clock.
3. Wait until RXNE=1 and read the SPI_DR register to get the received data (this clears the RXNE bit). Repeat this operation for each data item to be received.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edge of the RXNE flag.

Note: *If it is required to disable the SPI after the last transfer, follow the recommendation described in Section 21.2.8*

Fig 21.2-8 RXNE behavior in receive-only mode (BIDIRMODE=0 and RXONLY=1) in case of continuous transfers



Bidirectional receive procedure (BIDIMODE=1 and BIDIOE=0)

In this mode, the procedure is similar to the Receive-only mode procedure except that the BIDIMODE bit has to be set and the BIDIOE bit cleared in the SPI_CR2 register before enabling the SPI.

Continuous and discontinuous transfers

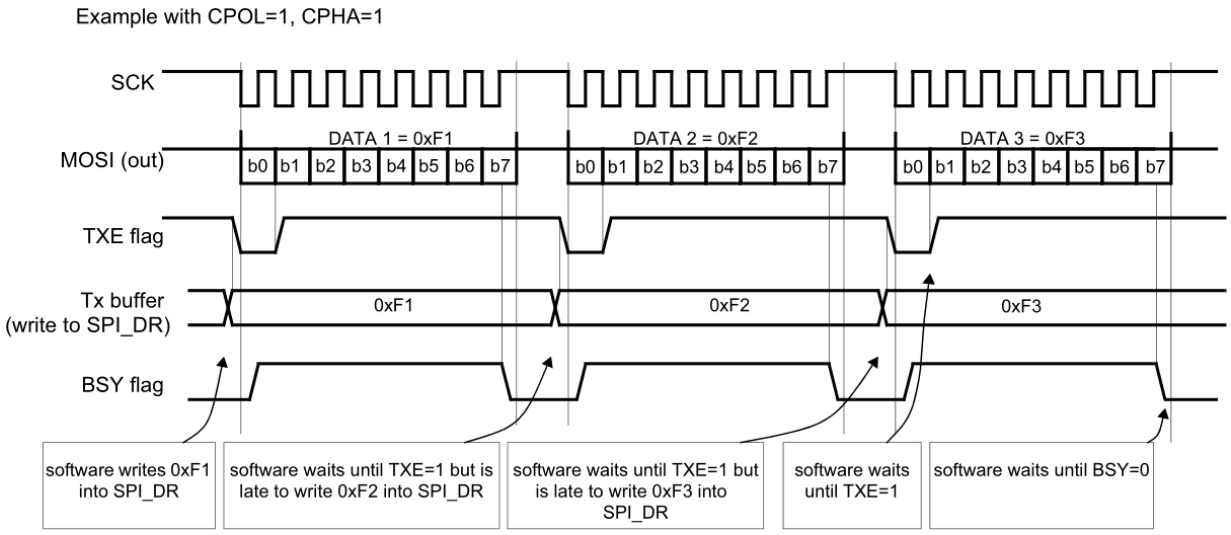
When transmitting data in master mode, if the software is fast enough to detect each rising edge of TXE (or TXE interrupt) and to immediately write to the SPI_DR register before the ongoing data transfer is complete, the communication is said to be continuous. In this case, there is no discontinuity in the generation of the SPI clock between each data item and the BSY bit is never cleared between each data transfer.

On the contrary, if the software is not fast enough, this can lead to some discontinuities in the communication. In this case, the BSY bit is cleared between each data transmission (see Figure 21.2-9).

In Master receive-only mode (RXONLY=1), the communication is always continuous and the BSY flag is always read at 1

In slave mode, the continuity of the communication is decided by the SPI master device. In any case, even if the communication is continuous, the BSY flag goes low between each transfer for a minimum duration of one SPI clock cycle (see Figure 21.2-8).

Fig 21.2-9 TXE/BSY behavior when transmitting (BIDIRMODE=0 and RXONLY=0) in case of discontinuous transfers



21.2.6 CRC calculation

A CRC calculator has been implemented for communication reliability. Separate CRC calculators are implemented for transmitted data and received data. The CRC is calculated using a programmable polynomial serially on each bit. It is calculated on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CR1 register.

Note: *This SPI offers two kinds of CRC calculation standard which depend directly on the data frame format selected for the transmission and/or reception: 8-bit data (CR8) and 16-bit data (CRC16).*

CRC calculation is enabled by setting the CRCEN bit in the SPI_CR1 register. This action resets the CRC registers (SPI_RXCRCR and SPI_TXCRCR). In full duplex or transmitter only mode, when the transfers are managed by the software (CPU mode), it is necessary to write the bit CRCNEXT immediately after the last data to be transferred is written to the SPI_DR. At the end of this last data transfer, the SPI_TXCRCR value is transmitted.

In receive only mode and when the transfers are managed by software (CPU mode), it is necessary to write the CRCNEXT bit after the second last data has been received. The CRC is received just after the last data reception and the CRC check is then performed.

At the end of data and CRC transfers, the CRCERR flag in the SPI_SR register is set if corruption occurs during the transfer.

If data are present in the TX buffer, the CRC value is transmitted only after the transmission of the data byte. During CRC transmission, the CRC calculator is switched off and the register value remains unchanged.

SPI communication using the CRC is possible through the following procedure:

1. Program the CPOL, CPHA, LSBFirst, BR, SSM, SSI and MSTR values.
2. Program the polynomial in the SPI_CRCPR register.
3. Enable the CRC calculation by setting the CRCEN bit in the SPI_CR1 register. This also clears the

SPI_RXCRCR and SPI_TXCRCR registers.

4. Enable the SPI by setting the SPE bit in the SPI_CR1 register.
5. Start the communication and sustain the communication until all but one byte or half- word have been transmitted or received.
   - In full duplex or transmitter-only mode, when the transfers are managed by software, when writing the last byte or half word to the Tx buffer, set the CRCNEXT bit in the SPI_CR1 register to indicate that the CRC will be transmitted after the transmission of the last byte.
   - In receiver only mode, set the bit CRCNEXT just after the reception of the second to last data to prepare the SPI to enter in CRC Phase at the end of the reception of the last data. CRC calculation is frozen during the CRC transfer.
6. After the transfer of the last byte or half word, the SPI enters the CRC transfer and check phase. In full duplex mode or receiver-only mode, the received CRC is compared to the SPI_RXCRCR value. If the value does not match, the CRCERR flag in SPI_SR is set and an interrupt can be generated when the ERRIE bit in the SPI_CR2 register is set.

Note:

When the SPI is in slave mode, be careful to enable CRC calculation only when the clock is stable, that is, when the clock is in the steady state. If not, a wrong CRC calculation may be done. In fact, the CRC is sensitive to the SCK slave input clock as soon as CRCEN is set, and this, whatever the value of the SPE bit.

With high bitrate frequencies, be careful when transmitting the CRC. As the number of used CPU cycles has to be as low as possible in the CRC transfer phase, it is forbidden to call software functions in the CRC transmission sequence to avoid errors in the last data and CRC reception. In fact, CRCNEXT bit has to be written before the end of the transmission/reception of the last data.

For high bit rate frequencies, it is advised to use the DMA mode to avoid the degradation of the SPI speed performance due to CPU accesses impacting the SPI bandwidth.

When the devices are configured as slaves and the NSS hardware mode is used, the NSS pin needs to be kept low between the data phase and the CRC phase.

When the SPI is configured in slave mode with the CRC feature enabled, CRC calculation takes place even if a high level is applied on the NSS pin. This may happen for example in case of a multislave environment where the communication master addresses slaves alternately.

Between a slave deselection (high level on NSS) and a new slave selection (low level on NSS), the CRC value should be cleared on both master and slave sides in order to resynchronize the master and slave for their respective CRC calculation.

To clear the CRC, follow the procedure below:

1. Disable SPI (SPE = 0)
2. Clear the CRCEN bit
3. Set the CRCEN bit
4. Enable the SPI (SPE = 1)

21.2.7 Status flags

Four status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

When it is set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can be loaded into the buffer. The TXE flag is cleared when writing to the SPI_DR register.

Rx buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the Rx buffer. It is cleared when SPI_DR is read.

BUSY flag

This BSY flag is set and cleared by hardware (writing to this flag has no effect). The BSY flag indicates the state of the communication layer of the SPI.

When BSY is set, it indicates that the SPI is busy communicating. There is one exception in master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software wants to disable the SPI and enter Halt mode (or disable the peripheral clock). This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is also useful to avoid write collisions in a multimaster system.

The BSY flag is set when a transfer starts, with the exception of master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0).

It is cleared:

- when a transfer is finished (except in master mode if the communication is continuous)
- when the SPI is disabled
- when a master mode fault occurs (MODF=1)

When communication is not continuous, the BSY flag is low between each communication.
When communication is continuous:

- in master mode, the BSY flag is kept high during all the transfers
- in slave mode, the BSY flag goes low for one SPI clock cycle between each transfer

Note: *Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.*

21.2.8 Disabling the SPI

When a transfer is terminated, the application can stop the communication by disabling the SPI peripheral. This is done by clearing the SPE bit.

For some configurations, disabling the SPI and entering the Halt mode while a transfer is ongoing can cause the current transfer to be corrupted and/or the BSY flag might become unreliable.

To avoid any of those effects, it is recommended to respect the following procedure when disabling the SPI:

In master or slave full-duplex mode (BIDIMODE=0, RXONLY=0)

1. Wait until RXNE=1 to receive the last data
2. Wait until TXE=1
3. Then wait until BSY=0
4. Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master or slave unidirectional transmit-only mode (BIDIMODE=0, RXONLY=0) or bidirectional transmit mode (BIDIMODE=1, BIDIOE=1)

After the last data is written into the SPI_DR register:

1. Wait until TXE=1
2. Then wait until BSY=0
3. Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master unidirectional receive-only mode (MSTR=1, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=1, BIDIMODE=1, BIDIOE=0)

This case must be managed in a particular way to ensure that the SPI does not initiate a new transfer:

1. Wait for the second to last occurrence of RXNE=1 (n–1)
2. Then wait for one SPI clock cycle (using a software loop) before disabling the SPI (SPE=0)
3. Then wait for the last RXNE=1 before entering the Halt mode (or disabling the peripheral clock)

Note: *In master bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0), the BSY flag is kept low during transfers.*

In slave receive-only mode (MSTR=0, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=0, BIDIMODE=1, BIDOE=0)

1. You can disable the SPI (write SPE=1) at any time: the current transfer will complete before the SPI is effectively disabled
2. Then, if you want to enter the Halt mode, you must first wait until BSY = 0 before entering the Halt mode (or disabling the peripheral clock)

21.2.9 SPI communication using DMA (direct memory addressing)

To operate at its maximum speed, the SPI needs to be fed with the data for transmission and the data received on the Rx buffer should be read to avoid overrun. To facilitate the transfers, the SPI features a DMA capability implementing a simple request/acknowledge protocol.

A DMA access is requested when the enable bit in the SPI_CR2 register is enabled. Separate requests must be issued to the Tx and Rx buffers (see Figure 21.2-10 and Figure 21.2-11):

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPI_DR register (this clears the TXE flag).

- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPI_DR register (this clears the RXNE flag).

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received are not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (flag TCIF is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until TXE=1 and then until BSY=0.

Note: *During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, it is mandatory to wait first until TXE=1 and then until BSY=0 after writing the last data.*

Fig 21.2-10 Transmission using DMA

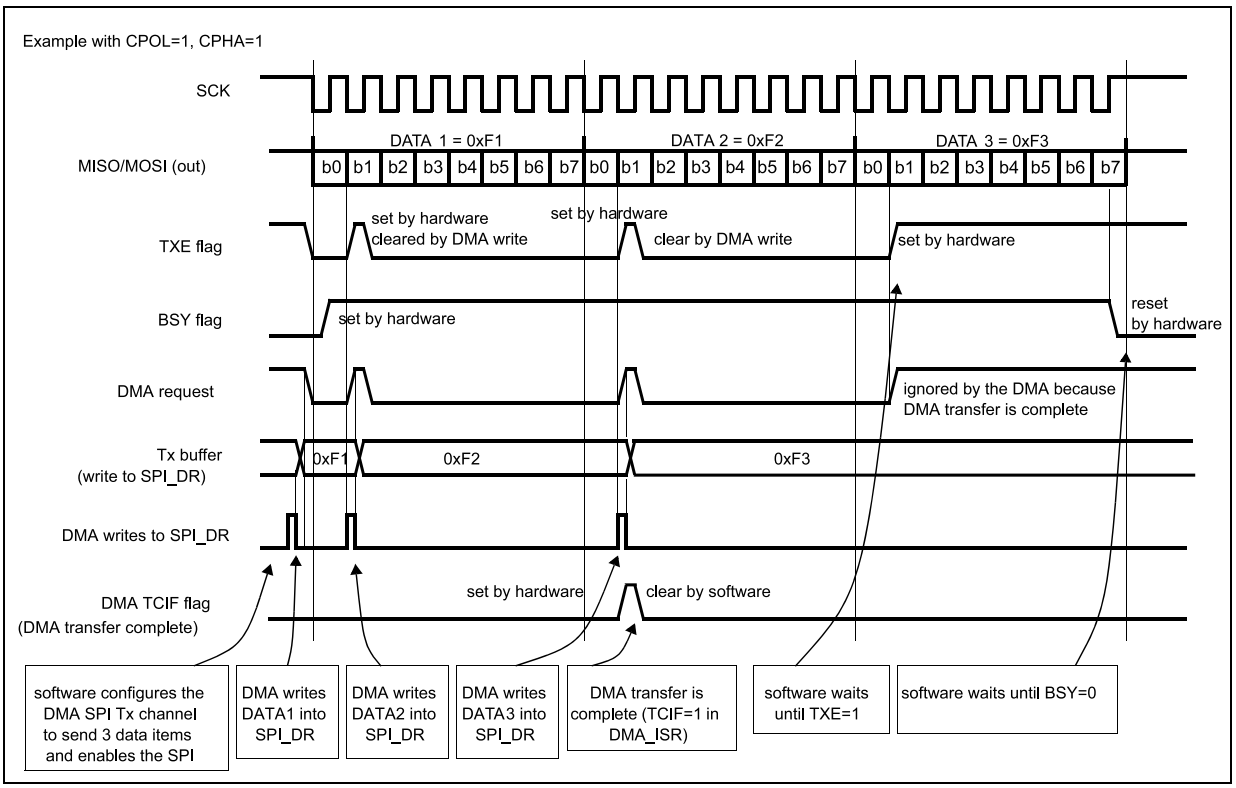
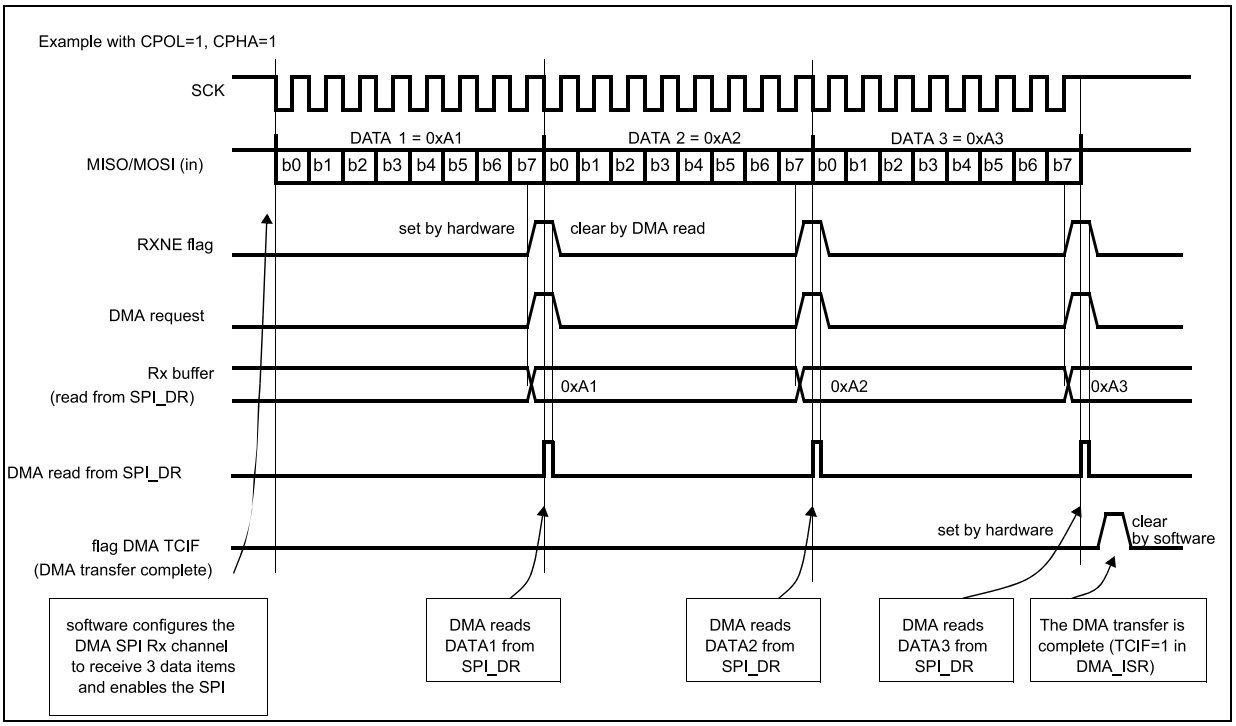


Fig 21.2-11 Reception using DMA



DMA capability with CRC

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication are automatic that is without using the bit CRCNEXT. After the CRC reception, the CRC must be read in the SPI_DR register to clear the RXNE flag.

At the end of data and CRC transfers, the CRCERR flag in SPI_SR is set if corruption occurs during the transfer.

21.2.10 Error flags

Master mode fault (MODF)

Master mode fault occurs when the master device has its NSS pin pulled low (in NSS hardware mode) or SSI bit low (in NSS software mode), this automatically sets the MODF bit. Master mode fault affects the SPI peripheral in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPI_SR register while the MODF bit is set.
2. Then write to the SPI_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original

state after this clearing sequence.

As a security, hardware does not allow the setting of the SPE and MSTR bits while the MODF bit is set.

In a slave device the MODF bit cannot be set. However, in a multimaster configuration, the device can be in slave mode with this MODF bit set. In this case, the MODF bit indicates that there might have been a multimaster conflict for system control. An interrupt routine can be used to recover cleanly from this state by performing a reset or returning to a default state.

Overrun condition

An overrun condition occurs when the master device has sent data bytes and the slave device has not cleared the RXNE bit resulting from the previous data byte transmitted. When an overrun condition occurs:

- the OVR bit is set and an interrupt is generated if the ERRIE bit is set.

In this case, the receiver buffer contents will not be updated with the newly received data from the master device. A read from the SPI_DR register returns this byte. All other subsequently transmitted bytes are lost.

Clearing the OVR bit is done by a read from the SPI_DR register followed by a read access to the SPI_SR register.

CRC error

This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CR1 register is set. The CRCERR flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRCR value.

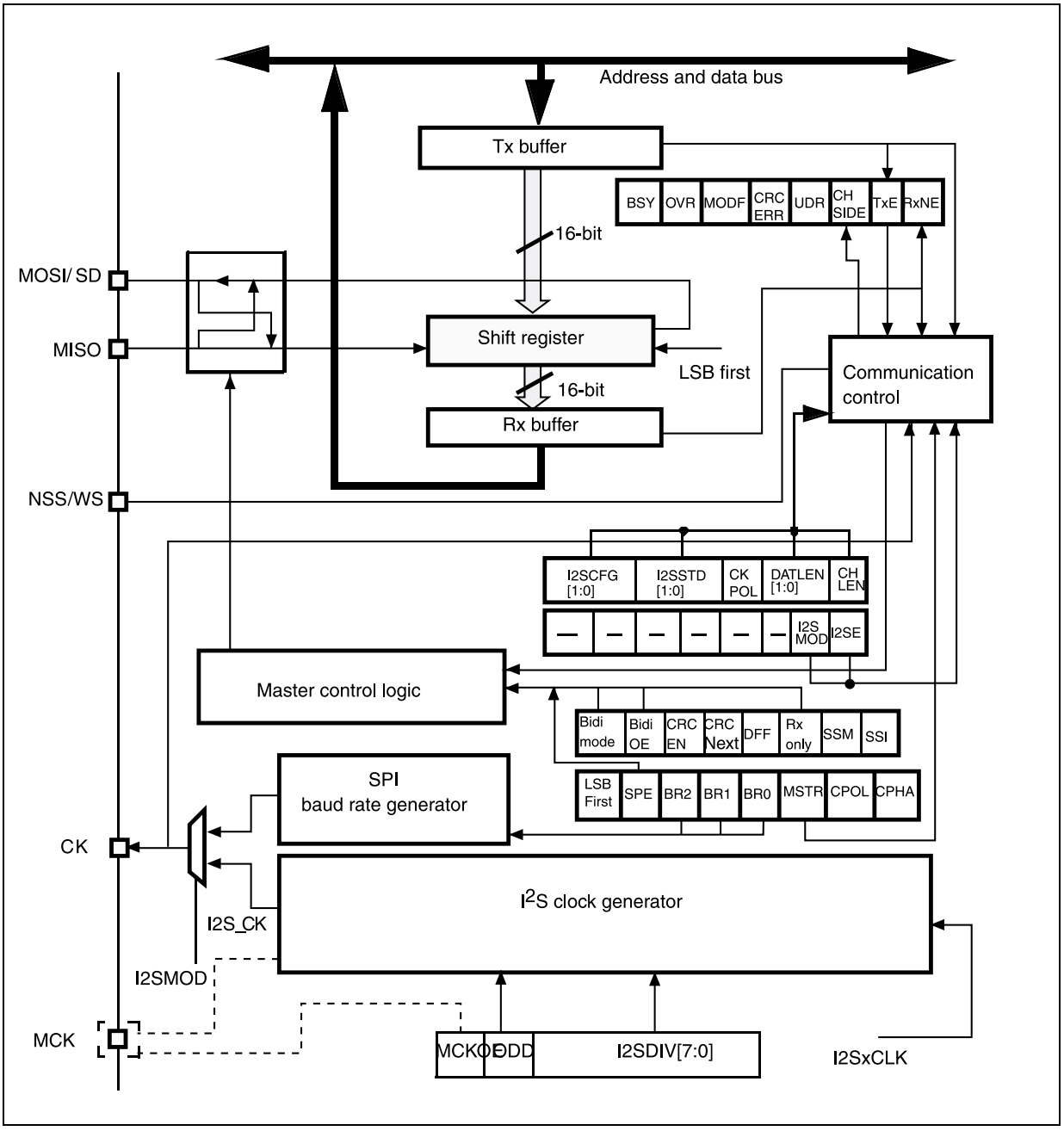
21.3 SPI interrupts

Interrupt event	Event flag	Enable Control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
CRC error flag	CRCERR	

21.4 I2S functional description

21.4.1 I2S general description

Fig 21.4-1 I2S block diagram



The SPI could function as an audio I2S interface when the I2S capability is enabled (by setting the I2SMOD bit in the SPI_I2SCFGR register). This interface uses almost the same pins, flags and interrupts as the SPI.

The I2S shares three common pins with the SPI:

- SD: Serial Data (mapped on the MOSI pin) to transmit or receive the two time- multiplexed data channels (in half-duplex mode only).

- WS: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin could be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I2S is configured in master mode (and when the MCKOE bit in the SPI_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to 256 × F_S , where F_S is the audio sampling frequency.

The I2S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I2S mode. One is linked to the clock generator configuration SPI_I2SPR and the other one is a generic I2S configuration register SPI_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.). The SPI_CR1 register and all CRC registers are not used in the I2S mode. Likewise, the SSOE bit in the SPI_CR2 register and the MODF and CRCERR bits in the SPI_SR are not used. The I2S uses the same SPI register for data transfer (SPI_DR) in 16-bit wide mode.

21.4.2 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for the transmission and the reception. So, it is up to the software to write into the data register the adequate value corresponding to the considered channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPI_SR register. Channel Left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol). Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in 16-bit frame
- 16-bit data packed in 32-bit frame
- 24-bit data packed in 32-bit frame
- 32-bit data packed in 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPI_DR or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 nonsignificant bits are extended to 32 bits with 0-bits (by hardware).

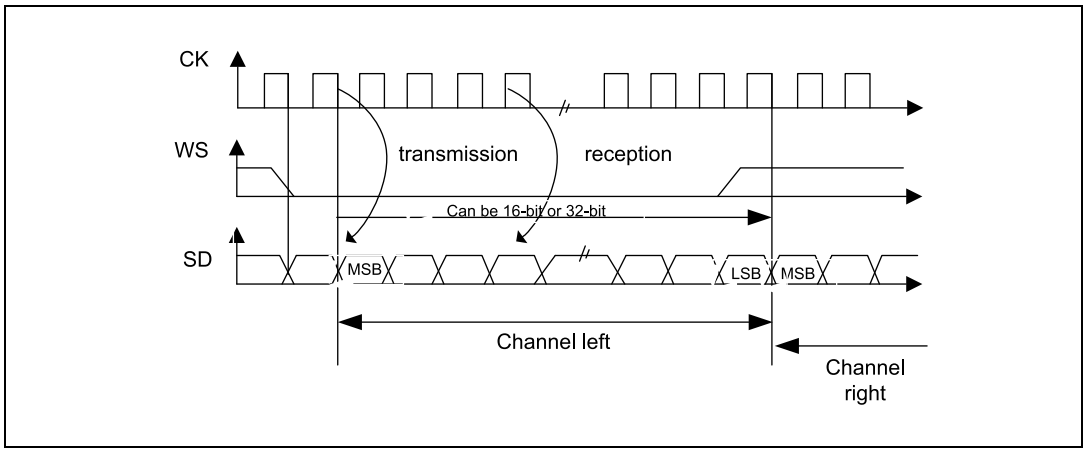
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I2S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPI_I2SCFGR register.

I2S Philips standard

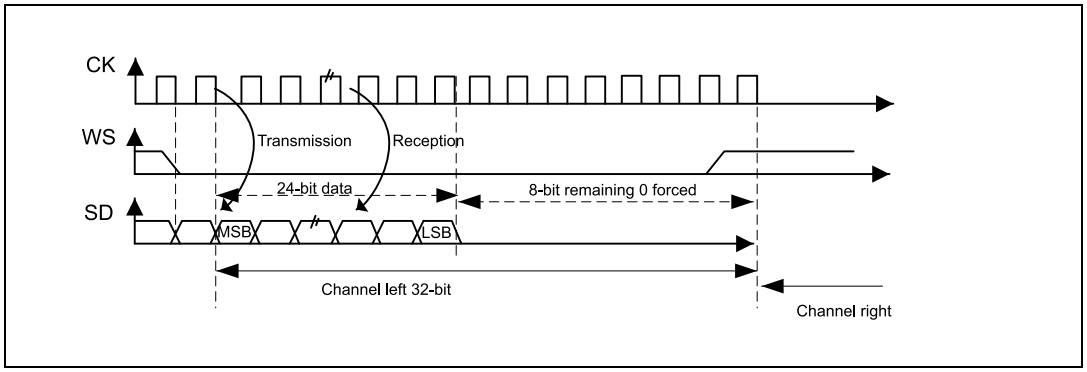
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

Fig 21.4-2 I2S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0)



Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

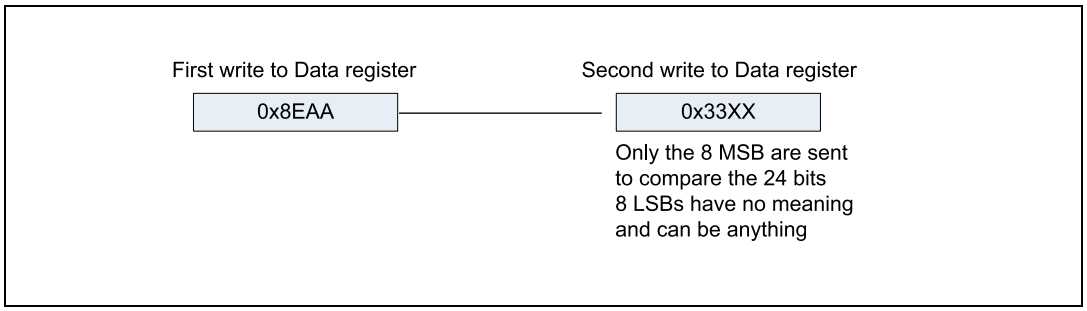
Fig 21.4-3 I2S Philips standard waveforms (24-bit frame with CPOL = 0)



This mode needs two write or read operations to/from the SPI_DR.

- In transmission mode:
  if 0x8EAA33 has to be sent (24-bit):

Fig 21.4-4 Transmitting 0x8EAA33



- In reception mode:
  if data 0x8EAA33 is received:

Fig 21.4-5 Receiving 0x8EAA33

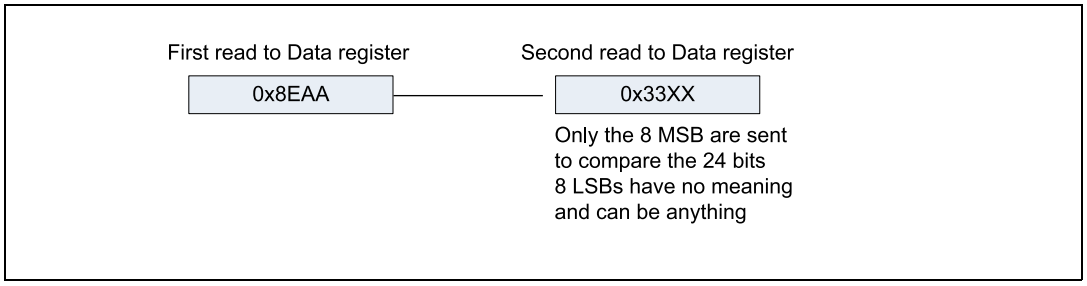
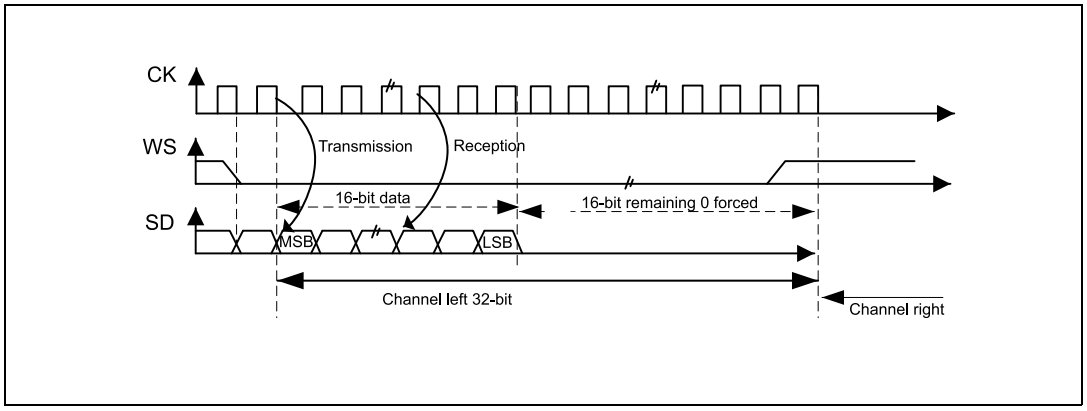


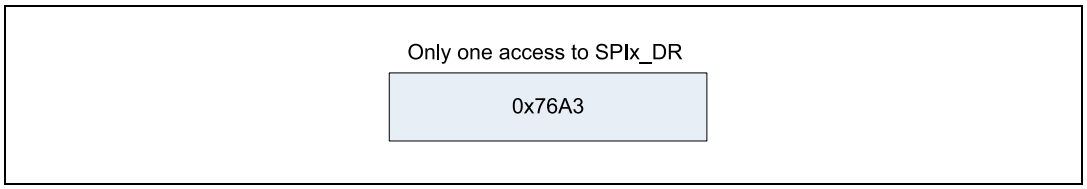
Fig 21.4-6 I2S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0)



When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, only one access to SPI_DR is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in Figure 21.4-9 is required.

Fig 21.4-7 Example



For transmission, each time an MSB is written to SPI_DR, the TXE flag is set and its interrupt, if allowed, is generated to load SPI_DR with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

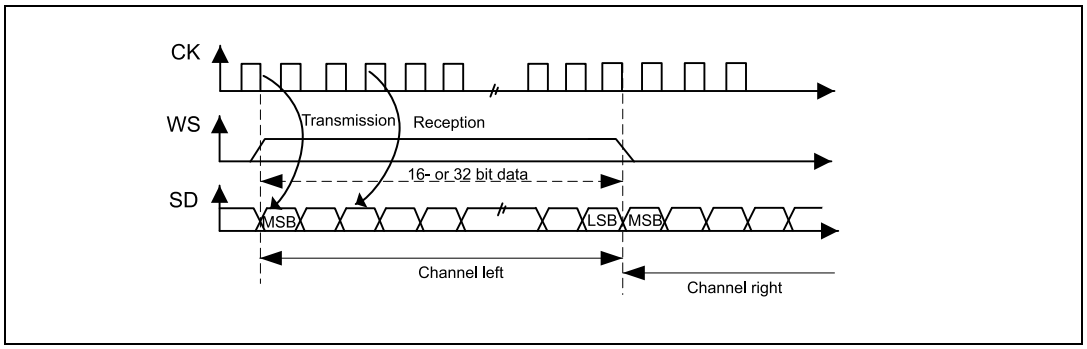
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

MSB justified standard

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

Fig 21.4-8 MSB justified 16-bit or 32-bit full-accuracy length with CPOL = 0



Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

Fig 21.4-9 MSB justified 24-bit frame length with CPOL = 0

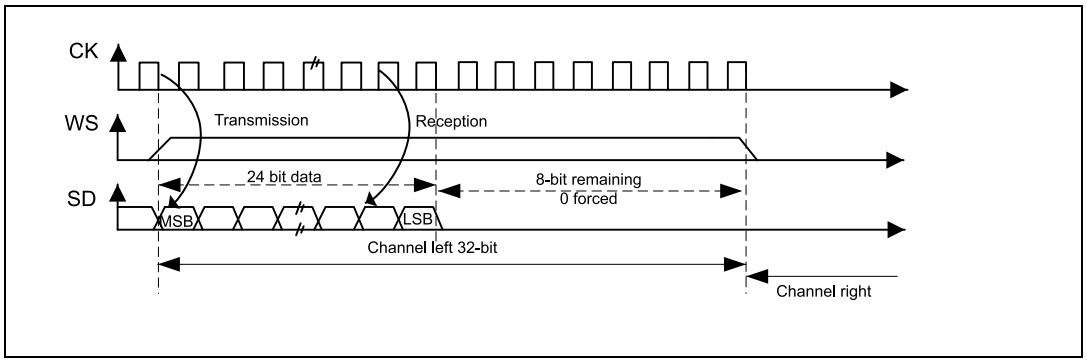


Fig 21.4-10 LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0

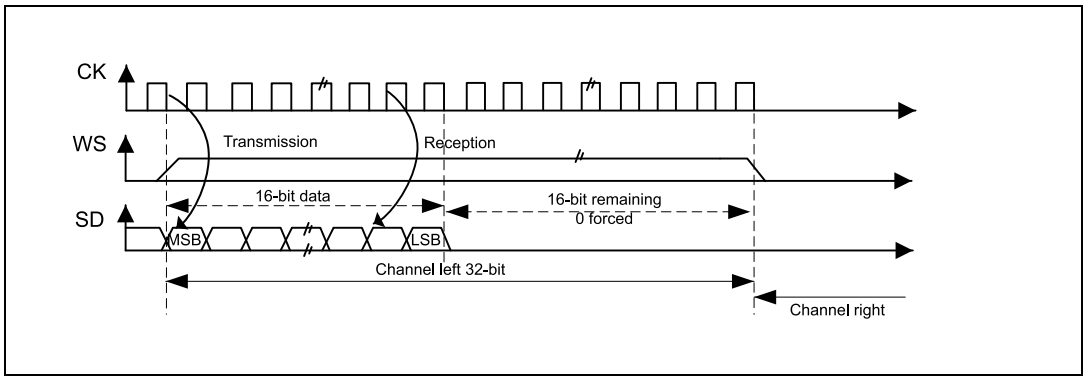
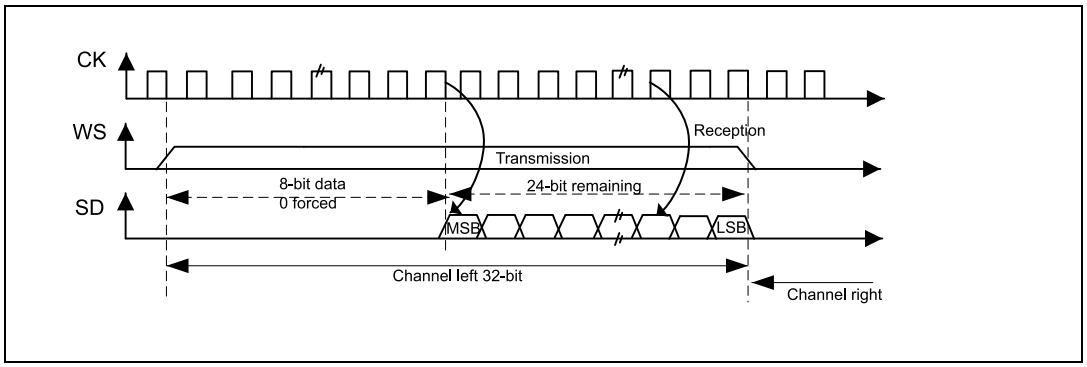
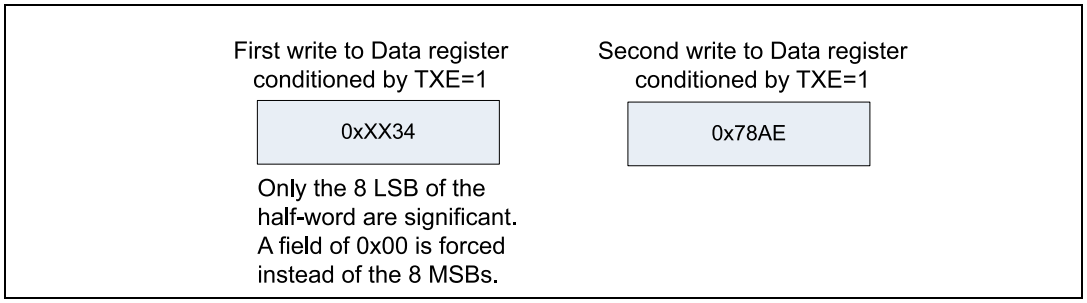


Fig 21.4-11 LSB justified 24-bit frame length with CPOL = 0



- In transmission mode:
  If data 0x3478AE have to be transmitted, two write operations to the SPI_DR register are required from software or by DMA. The operations are shown below.

Fig 21.4-12 Operations required to transmit 0x3478AE



- In reception mode:
  If data 0x3478AE are received, two successive read operations from SPI_DR are required on each RXNE event.

Fig 21.4-13 Operations required to receive 0x3478AE

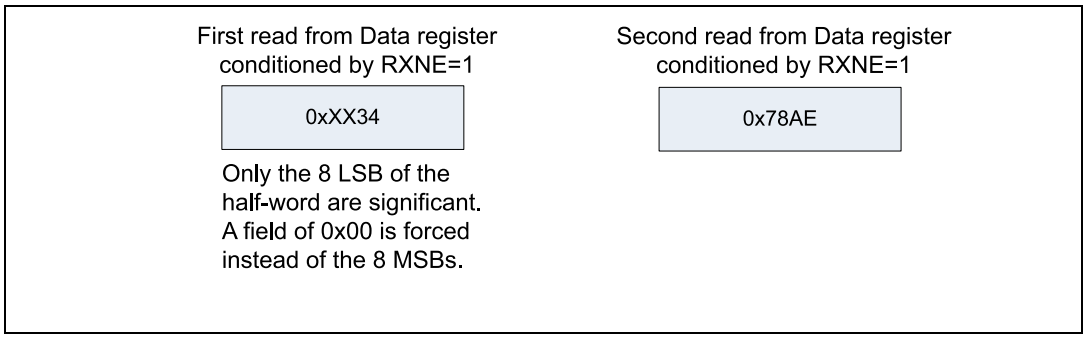
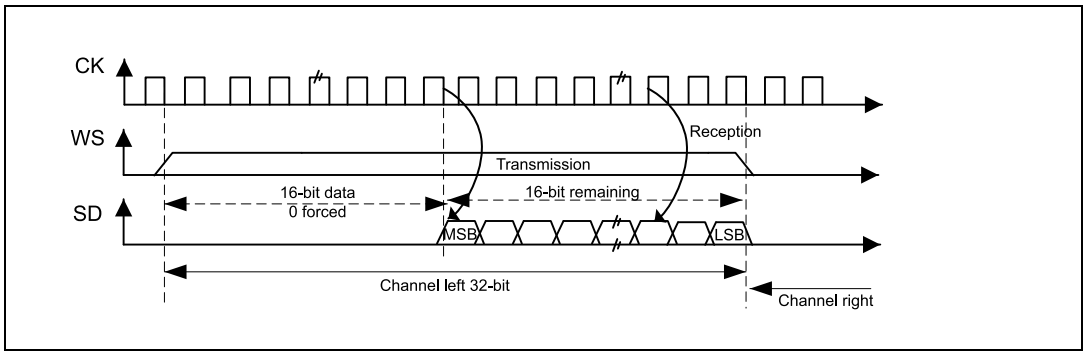


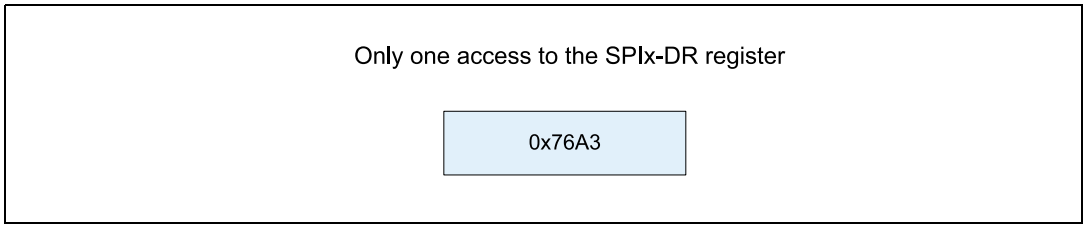
Fig 21.4-14 LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0



When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, Only one access to SPI_DR is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in Figure 264 is required.

Fig 21.4-15 Example of LSB justified 16-bit extended to 32-bit packet frame



In transmission mode, when TXE is asserted, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). TXE is asserted again as soon as the effective data (0x76A3) is sent on SD.

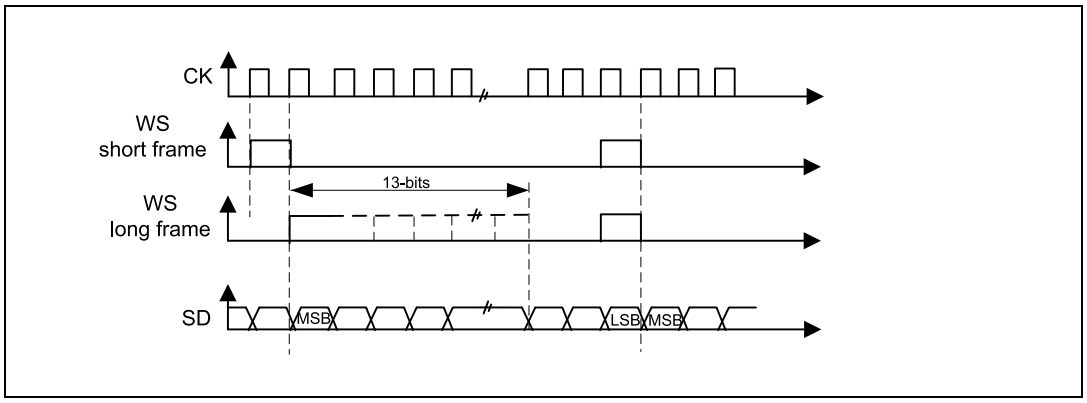
In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

PCM standard

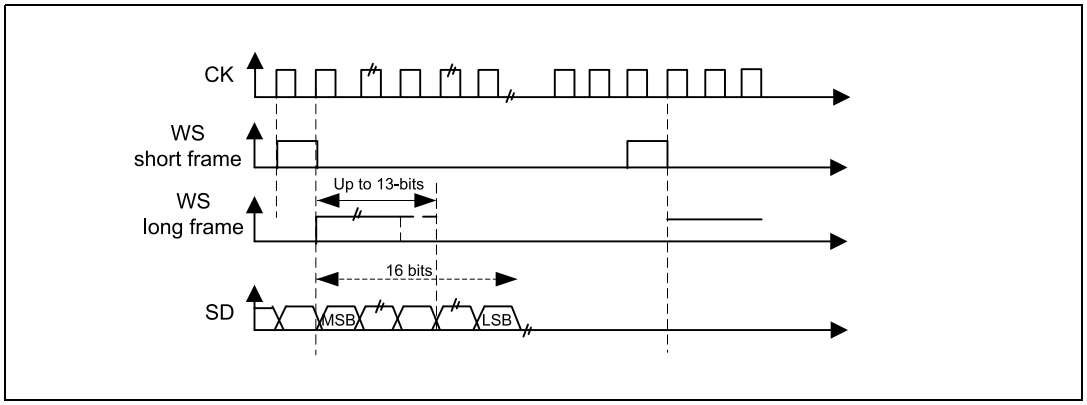
For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPI_I2SCFGR.

Fig 21.4-16 PCM standard waveforms (16-bit)



For long frame synchronization, the WS signal assertion time is fixed 13 bits in master mode. For short frame synchronization, the WS synchronization signal is only one cycle long.

Fig 21.4-17 PCM standard waveforms (16-bit extended to 32-bit packet frame)



Note: *For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPI_I2SCFGR register) even in slave mode.*

Clock generator

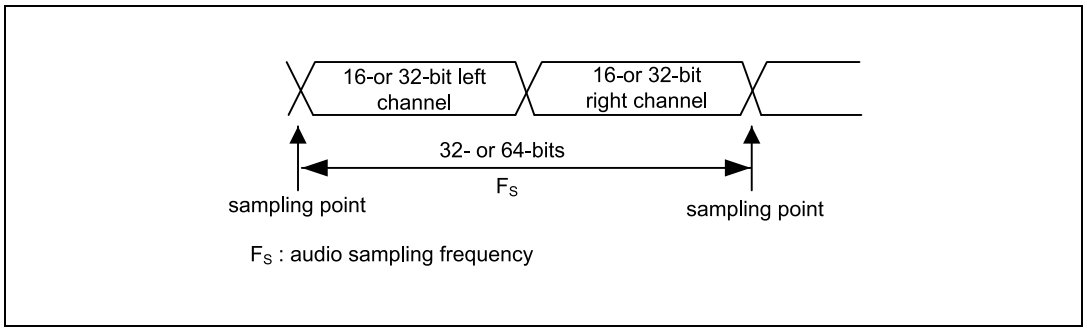
The I2S bitrate determines the dataflow on the I2S data line and the I2S clock signal frequency.

I2S bitrate = number of bits per channel × number of channels × sampling audio frequency For a 16-bit audio, left and right channel, the I2S bitrate is calculated as follows:

$$\text{I2S bitrate} = 16 \times 2 \times F_S$$

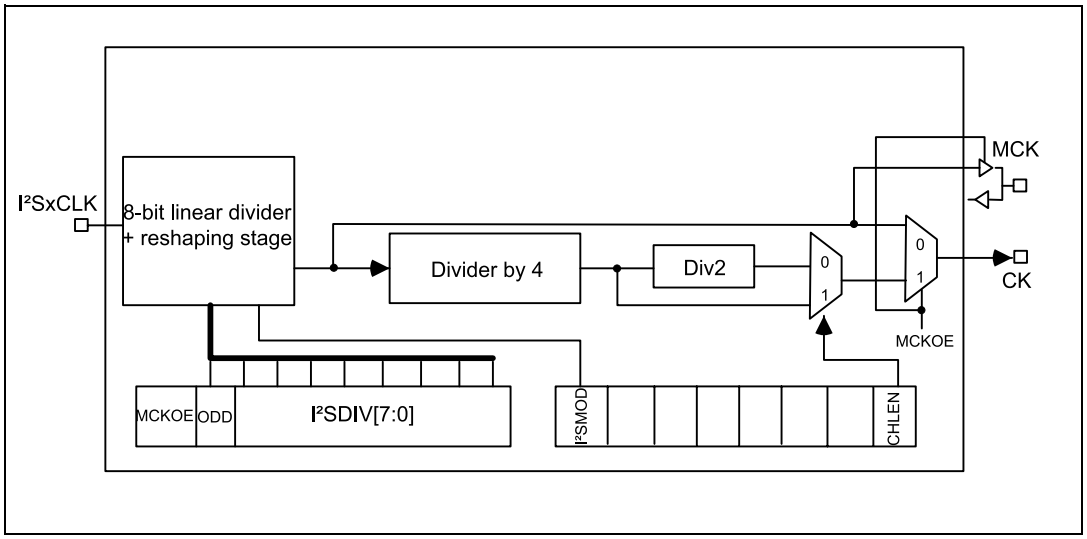
It will be: I2S bitrate = 32 x 2 x F_S if the packet length is 32-bit wide

Fig 21.4-18 Audio sampling frequency definition



When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency

Fig 21.4-19 I2S clock generator architecture



1. Where x could be 2 or 3.

Figure 21.4-18 presents the communication clock architecture. . The I2SxCLK source is the system clock (provided by the HSI, the HSE or the PLL, and sourcing the AHB clock). For connectivity line devices, tThe I2SxCLK source can be either SYSCLK or the PLL3 VCO (2 × PLL3CLK) clock in order to achieve the maximum accuracy. This selection is made using the I2S2SRC and I2S3SRC bits in the RCC_CFGR2 register.

The audio sampling frequency can be 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range). In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

When the master clock is generated (MCKOE in the SPI_I2SPR register is set):

- F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)*8] when the channel frame is 16-bit wide
- F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)*4] when the channel frame is 32-bit wide

When the master clock is disabled (MCKOE bit cleared):

- F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)] when the channel frame is 16-bit wide
- F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)] when the channel frame is 32-bit wide

Table:21.4-1 provide example precision values for different clock configurations.

Note: *Other configurations are possible that allow optimum clock precision.*

Tab 21.4-1 Audio-frequency precision using standard 8 MHz HSE

SYSCLK(MHz)	I2S_DIV		I2S_ODD		MCLK	Target f_S(Hz)	Real f_S(KHz)		Error	
	16-bit	32-bit	16-bit	32-bit			16-bit	32-bit	16-bit	32-bit
72	11	6	1	0	No	96000	97826.09	93750	1.90%	2.34%
72	23	11	1	1	No	48000	47872.34	48913.04	0.27%	1.90%
72	25	13	1	0	No	44100	44117.65	43269.23	0.04%	1.88%
72	35	17	0	1	No	32000	32142.86	32142.86	0.44%	0.44%
72	51	25	0	1	No	22050	22058.82	22058.82	0.04%	0.04%
72	70	35	1	0	No	16000	15675.75	16071.43	0.27%	0.45%
72	102	51	0	0	No	11025	11029.41	11029.41	0.04%	0.04%
72	140	70	1	1	No	8000	8007.11	7978.72	0.09%	0.27%
72	2	2	0	0	Yes	96000	70312.15	70312.15	26.76%	26.76%
72	3	3	0	0	Yes	48000	46875	46875	2.34%	2.34%
72	3	3	0	0	Yes	44100	46875	46875	6.29%	6.29%
72	4	4	1	1	Yes	32000	31250	31250	2.34%	2.34%
72	6	6	1	1	Yes	22050	21634.61	21634.61	1.88%	1.88%
72	9	9	0	0	Yes	16000	15625	15625	2.34%	2.34%
72	13	13	0	0	Yes	11025	10817.30	10817.30	1.88%	1.88%
72	17	17	1	1	Yes	8000	8035.71	8035.71	0.45%	0.45%

21.4.3 I2S master mode

The I2S can be configured in master mode for transmission and reception. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, thanks to the MCKOE bit in the SPI_I2SPR register.

21.4.4 Procedure

- Select the I2SDIV[7:0] bits in the SPI_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPI_I2SPR register also has to be defined.
- Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPI_I2SPR register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to Section 21.4.2).
- Set the I2SMOD bit in SPI_I2SCFGR to activate the I2S functionalities and choose the I2S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the CHLEN bit. Select also the I2S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPI_I2SCFGR register.
- If needed, select all the potential interruption sources and the DMA capabilities by writing the SPI_CR2 register.

- The I2SE bit in SPI_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPI_I2SPR is set.

Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer. Assumedly, the first data written into the Tx buffer correspond to the channel Left data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the channel Right have to be written into the Tx buffer. The CHSIDE flag indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high. A full frame has to be considered as a Left channel data transmission followed by a Right channel data transmission. It is not possible to have a partial frame where only the left channel is sent. The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set. For more details about the write operations depending on the I2S standard mode selected, refer to Section 21.4.2 . To ensure a continuous audio data transmission, it is mandatory to write the SPI_DR with the next data to transmit before the end of the current transmission. To switch off the I2S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 3 (refer to the procedure described in Section 21.4.3), where the configuration should set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPI_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPI_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I2S cell.

For more details about the read operations depending on the I2S standard mode selected, refer to Section 21.4.2.

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPI_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S, specific actions are required to ensure that the I2S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB

justified mode (I2SSTD = 10)

1. Wait for the second to last RXNE = 1 (n –1)
2. Then wait 17 I2S clock cycles (using a software loop)
3. Disable the I2S (I2SE = 0)

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, I2S or PCM modes (I2SSTD = 00, I2SSTD = 01 or I2SSTD = 11, respectively)
  1. Wait for the last RXNE
  2. Then wait 1 I2S clock cycle (using a software loop)
  3. Disable the I2S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I2S:
  1. Wait for the second to last RXNE = 1 (n –1)
  2. Then wait one I2S clock cycle (using a software loop)
  3. Disable the I2S (I2SE = 0)

Note: *The BSY flag is kept low during transfers.*

21.4.5 I2S slave mode

In slave mode, the I2S can be configured in transmission or reception mode.The operating mode is following mainly the same rules as described for the I2S master configuration. In slave mode, there is no clock to be generated by the I2S interface. The clock and WS signals are input from the external master connected to the I2S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPI_I2SCFGR register to reach the I2S functionalities and choose the I2S standard through the I2SSTD[1:0] bits, the data length through the DATLEN[1:0] bits and the number of bits per channel for the frame configuring the CHLEN bit. Select also the mode (transmission or reception) for the slave through the I2SCFG[1:0] bits in SPI_I2SCFGR register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPI_CR2 register.
3. The I2SE bit in SPI_I2SCFGR register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I2S data register has to be loaded before the master initiates the communication.

For the I2S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I2S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission

mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

Note: *The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.*

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Note that the TXE flag should be checked to be at 1 before attempting to write the Tx buffer. For more details about the write operations depending on the I2S standard mode selected, refer to Section 21.4.2

To secure a continuous audio data transmission, it is mandatory to write the SPI_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPI_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPI_CR2 register, an interrupt is generated when the UDR flag in the SPI_SR register goes high. In this case, it is mandatory to switch off the I2S and to restart a data transfer starting from the left channel.

To switch off the I2S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in Section 21.4.5), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPI_I2SCFGR register.

Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPI_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer.

The CHSIDE flag is updated each time data are received to be read from SPI_DR. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPI_DR register.

For more details about the read operations depending the I2S standard mode selected, refer to Section 21.4.2

If data are received while the precedent received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPI_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

Note: *The external master components should have the capability of sending/receiving data in 16- bit or 32-bit packets via an audio channel*

21.4.6 Status flags

Three status flags are provided for the application to fully monitor the state of the I2S bus.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). It indicates the state of the communication layer of the I2S.

When BSY is set, it indicates that the I2S is busy communicating. There is one exception in master receive mode (I2SCFG = 11) where the BSY flag is kept low during reception. The BSY flag is useful to detect the end of a transfer if the software needs to disable the I2S. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is set when a transfer starts, except when the I2S is in master receiver mode.

The BSY flag is cleared:

- when a transfer completes (except in master transmit mode, in which the communication is supposed to be continuous)
- when the I2S is disabled

When communication is continuous:

- In master transmit mode, the BSY flag is kept high during all the transfers
- In slave mode, the BSY flag goes low for one I^2S clock cycle between each transfer

Note: *Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.*

Tx buffer empty flag (TXE)

When set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can then be loaded into it. The TXE flag is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I2S is disabled (I2SE bit is reset).

RX buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when SPI_DR register is read.

Channel Side flag (CHSIDE)

In transmission mode, this flag is refreshed when TXE goes high. It indicates the channel side to which the data to transfer on SD has to belong. In case of an underrun error event in slave transmission mode, this flag is not reliable and I2S needs to be switched off and switched on before resuming the communication.

In reception mode, this flag is refreshed when data are received into SPI_DR. It indicates from which channel side data have been received. Note that in case of error (like OVR) this flag becomes meaningless and the I2S should be reset by disabling and then enabling it (with configuration if it needs changing).

This flag has no meaning in the PCM standard (for both Short and Long frame modes). When the OVR or UDR flag in the SPI_SR is set and the ERRIE bit in SPI_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPI_SR status register (once the interrupt source has been cleared).

21.4.7 Error flags

There are two error flags for the I2S cell.

Underrun flag (UDR)

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPI_DR. It is available when the I2SMOD bit in SPI_I2SCFGR is set. An interrupt may be generated if the ERRIE bit in SPI_CR2 is set.
The UDR bit is cleared by a read operation on the SPI_SR register.

Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from SPI_DR. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in SPI_CR2.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPI_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPI_DR register followed by a read access to the SPI_SR register.

21.4.8 I2S interrupts

Tab 21.4-2 I2S interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Overrun error	OVR	ERRIE
Underrun error	UDR	

21.4.9 DMA features

DMA is working in exactly the same way as for the SPI mode. There is no difference on the I2S. Only the CRC feature is not available in I2S mode since there is no data transfer protection system

21.5 SPI and I2S registers

The peripheral registers have to be accessed by half-words (16 bits) or words (32 bits).

Tab 21.5-1 SPI and I2S registers map

offset	Register	Reset value	Description
SPI3/I2S3 bass address: = 0x4000_3C00 SPI2/I2S3 bass address: = 0x4000_3800			
0x00	SPI_CR1	0x0000_0000	SPI control register 1 （not used in I2S mode）
0x04	SPI_CR2	0x0000_0000	SPI control register 2
0x08	SPI_SR	0x0000_0002	SPI status register
0x0C	SPI_DR	0x0000_0000	SPI data register
0x10	SPI_CRCPR	0x0000_0007	SPI CRC polynomial register （not used in I2S mode）
0x14	SPI_RXCRCR	0x0000_0000	SPI RX CRC register （not used in I2S mode）
0x18	SPI_TXCRCR	0x0000_0000	SPI TX CRC register（not used in I2S mode）
0x1C	SPI_I2SCFGR	0x0000_0000	SPI_I2S configuration register
0x20	SPI_I2SPR	0x0000_0002	SPI_I2S prescaler register

21.5.1 SPI control register 1(SPI_CR1)

Note: *not used in I2S mode*

Register	Address offset	Access	Reset value	Description
SPI_CR1	0x00	RW	0x0000_0000	SPI control register 1

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI
7	6	5	4	3	2	1	0
LSBFIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA

SPI control register 1(SPI_CR1)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15]	RW	**BIDIMODE:** Bidirectional data mode enable 0: 2-line unidirectional data mode selected 1: 1-line bidirectional data mode selected
[14]	RW	**BIDIOE:** Output enable in bidirectional mode This bit combined with the BIDImode bit selects the direction of transfer in bidirectional mode 0: Output disabled (receive-only mode) 1: Output enabled (transmit-only mode) **Note:** *In master mode, the MOSI pin is used while the MISO pin is used in slave mode.*

[13]	RW	**CRCEN:** Hardware CRC calculation enable 0: CRC calculation disabled 1: CRC calculation enabled **Note:** *This bit should be written only when SPI is disabled (SPE = '0') for correct operation.*
[12]	RW	**CRCNEXT:** CRC transfer next 0: Data phase (no CRC phase) 1: Next transfer is CRC (CRC phase) **Note:** *When the SPI is configured in full duplex or transmitter only modes, CRCNEXT must be written as soon as the last data is written to the SPI_DR register.* *When the SPI is configured in receiver only mode, CRCNEXT must be set after the second last data reception.* *This bit should be kept cleared when the transfers are managed by DMA*
[11]	RW	**DFF:** Data frame format 0: 8-bit data frame format is selected for transmission/reception 1: 16-bit data frame format is selected for transmission/reception **Note:** *This bit should be written only when SPI is disabled (SPE = '0') for correct operation.*
[10]	RW	**RXONLY:** Receive only This bit combined with the BIDImode bit selects the direction of transfer in 2-line unidirectional mode. This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted. 0: Full duplex (Transmit and receive) 1: Output disabled (Receive-only mode)
[9]	RW	**SSM:** Software slave management When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit. 0: Software slave management disabled 1: Software slave management enabled
[8]	RW	**SSI:** Internal slave select This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the IO value of the NSS pin is ignored
[7]	RW	**LSBFIRST:** Frame format 0: MSB transmitted first 1: LSB transmitted first **Note:** *This bit should not be changed when communication is ongoing.*
[6]	RW	**SPE:** SPI enable 0: Peripheral disabled 1: Peripheral enable **Note:** *When disabling the SPI, follow the procedure described in Section 21.2.8*
[5:3]	RW	**BR[2:0]:** Baud rate control 000: $f_{PCLK}/2$ 001: $f_{PCLK}/4$ 010: $f_{PCLK}/8$ 011: $f_{PCLK}/16$ 100: $f_{PCLK}/32$ 101: $f_{PCLK}/64$ 110: $f_{PCLK}/128$ 111: $f_{PCLK}/256$ **Note:** *This bit should not be changed when communication is ongoing.*
[2]	RW	**MSTR:** Master selection 0: Slave configuration 1: Master configuration **Note:** *This bit should not be changed when communication is ongoing.*

[1]	RW	**CPOL:** Clock polarity 0: CK to 0 when idle 1: CK to 1 when idle **Note:** *This bit should not be changed when communication is ongoing.*
[0]	RW	**CPHA:** Clock phase 0: The first clock transition is the first data capture edge 1: The second clock transition is the first data capture edge **Note:** *This bit should not be changed when communication is ongoing.*

21.5.2 SPI control register 2(SPI_CR2)

Register	Address offset	Access	Reset value	Description
SPI_CR2	0x04	RW	0x0000_0000	SPI control register 2

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

SPI control register 2(SPI_CR2)bit description

Bit	Access	Description
[31:8]	R	Reserved, must be kept at reset value.
[7]	RW	**TXEIE:** Tx buffer empty interrupt enable 0: TXE interrupt masked 1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.
[6]	RW	**RXNEIE:** RX buffer not empty interrupt enable 0: RXNE interrupt masked 1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set
[5]	RW	**ERRIR:** Error interrupt enable This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode and UDR, OVR in I2S mode). 0: Error interrupt is masked 1: Error interrupt is enabled
[4:3]	R	Reserved, must be kept at reset value.
[2]	RW	**SSOE:** SS output enable 0: SS output is disabled in master mode and the cell can work in multimaster configuration 1: SS output is enabled in master mode and when the cell is enabled. The cell cannot work in a multimaster environment **Note:** *This bit is not used in I2S mode*

[1]	RW	**TXDMAEN:** Tx buffer DMA enable When this bit is set, the DMA request is made whenever the TXE flag is set. 0: Tx buffer DMA disabled 1: Tx buffer DMA enabled
[0]	RW	**RXDMAEN:** Rx buffer DMA enable When this bit is set, the DMA request is made whenever the RXNE flag is set. 0: Rx buffer DMA disabled 1: Rx buffer DMA enabled

21.5.3 SPI status register(SPI_SR)

Register	Address offset	Access	Reset value	Description
SPI_SR	0x08	R	0x0000_0002	SPI status register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
BYS	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE

SPI status register(SPI_SR)bit description

Bit	Access	Description
[31:7]	R	Reserved, must be kept at reset value.
[7]	R	**BSY:** Busy flag 0: SPI (or I2S) not busy 1: SPI (or I2S) is busy in communication or Tx buffer is not empty This flag is set and cleared by hardware **Note:** *BSY flag must be used with caution: refer to Section 21.2.7 and Section 21.2.8*
[6]	R	**OVR:** Overrun flag 0: No overrun occurred 1: Overrun occurred This flag is set by hardware and reset by a software sequence. Refer to Section 21.4.7 for the software sequence.
[5]	R	**MODF:** Mode fault 0: No mode fault occurred 1: Mode fault occurred This flag is set by hardware and reset by a software sequence. Refer to Section 21.2.10 for the software sequence. **Note:** *This bit is not used in I2S mode*
[4]	RC_W0	**CRCERR:** CRC error flag 0: CRC value received matches the SPI_RXCRCR value 1: CRC value received does not match the SPI_RXCRCR value This flag is set by hardware and cleared by software writing 0. **Note:** *This bit is not used in I2S mode*

[3]	R	**UDR:** Underrun flag 0: No underrun occurred 1: Underrun occurred This flag is set by hardware and reset by a software sequence. Refer to Section 21.4.7 for the software sequence. **Note:** *This bit is not used in SPI mode*
[2]	R	**CHSIDE:** Channel side 0: Channel Left has to be transmitted or has been received 1: Channel Right has to be transmitted or has been received **Note:** *This bit is not used for SPI mode and is meaningless in PCM mode.*
[1]	R	**TXE:** Transmit buffer empty 0: Tx buffer not empty 1: Tx buffer empty
[0]	R	**RXNE:** Receive buffer not empty 0: Rx buffer empty 1: Rx buffer not empty

21.5.4 SPI data register(SPI_DR)

Register	Address offset	Access	Reset value	Description
SPI_DR	0x0C	RW	0x0000_0000	SPI data register

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

SPI data register(SPI_DR)bit description

Bit	Access	Description
[31:16]	R	Reserved

[15:0]	RW	**DR[15:0]:** Data register Data received or to be transmitted. The data register is split into 2 buffers - one for writing (Transmit Buffer) and another one for reading (Receive buffer). A write to the data register will write into the Tx buffer and a read from the data register will return the value held in the Rx buffer **Note:** *These notes apply to SPI mode:* *Depending on the data frame format selection bit (DFF in SPI_CR1 register), the data sent or received is either 8-bit or 16-bit. This selection has to be made before enabling the SPI to ensure correct operation.* *For an 8-bit data frame, the buffers are 8-bit and only the LSB of the register (SPI_DR[7:0]) is used for transmission/reception. When in reception mode, the MSB of the register (SPI_DR[15:8]) is forced to 0.* *For a 16-bit data frame, the buffers are 16-bit and the entire register, SPI_DR[15:0] is used for transmission/reception.*

21.5.5 SPI CRC polynomial register(SPI_CRCPR)

Note: *Not used in I2S mode*

Register	Address offset	Access	Reset value	Description
SPI_CRCPR	0x10	RW	0x0000_0007	SPI CRC polynomial register

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

SPI CRC polynomial register(SPI_CRCPR)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**CRCPOLY[15:0]:** CRC polynomial register This register contains the polynomial for the CRC calculation. The CRC polynomial (0007h) is the reset value of this register. Another polynomial can be configured as required. **Note:** *hese bits are not used for the I2S mode*

21.5.6 SPI RX CRC register(SPI_RXCRCR)

Note: *Not used in I2S mode*

Register	Address offset	Access	Reset value	Description
SPI_RXCRCR	0x14	R	0x0000_0000	SPI RX CRC register

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

SPI RX CRC register(SPI_RXCRCR)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	R	**RXCRC[15:0]:** Rx CRC register When CRC calculation is enabled, the RxCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPI_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register. Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_CR1 is cleared). CRC calculation is done based on any CRC8 standard. The entire 16-bits of this register are considered when a 16-bit data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard. **Note:** *A read to this register when the BSY Flag is set could return an incorrect value. These bits are not used for I2S mode.*

21.5.7 SPI_TX CRC register(SPI_TXCRCR)

Note: *Not used in I2S mode*

Register	Address offset	Access	Reset value	Description
SPI_TXCRCR	0x18	R	0x0000_0000	SPI_TX CRC register

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

SPI_TX CRC register(SPI_RXCRCR)bit description

Bit	Access	Description
[31:16]	R	Reserved

[15:0]	R	**TXCRC[15:0]:** Tx CRC register When CRC calculation is enabled, the TxCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPI_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register. Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_CR1 is cleared). CRC calculation is done based on any CRC8 standard. The entire 16-bits of this register are considered when a 16-bit data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard. **Note:** *A read to this register when the BSY flag is set could return an incorrect value. These bits are not used for I2S mode.*

21.5.8 SPI_I2S configuration register(SPI_I2SCFGR)

Register	Address offset	Access	Reset value	Description
SPI_I2SCFGR	0x1C	R	0x0000_0000	SPI_I2S configuration register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				I2SMOD	I2SE	I2SCFG[1:0]	
7	6	5	4	3	2	1	0
PCMSYNC	Reserved	I2SSTD[1:0]		CKPOL	DATLEN[1:0]		CHLEN

SPI_I2S configuration register(SPI_I2SCFGR)bit description

Bit	Access	Description
[31:12]	R	Reserved, must be kept at reset value.
[11]	RW	**I2SMOD:** I2S mode selection 0: SPI mode is selected 1: I2S mode is selected **Note:** *This bit should be configured when the SPI or I2S is disabled*
[10]	RW	**I2SE:** I2S Enable 0: I2S peripheral is disabled 1: I2S peripheral is enabled **Note:** *This bit is not used in SPI mode.*
[9:8]	RW	**I2SCFG[1:0]:** I2S configuration mode 00: Slave - transmit 01: Slave - receive 10: Master - transmit 11: Master - receive **Note:** *This bit should be configured when the I2S is disabled. It is not used in SPI mode.*

[7]	RW	**PCMSYNC:** PCM frame synchronization 0: Short frame synchronization 1: Long frame synchronization **Note:** *This bit has a meaning only if I2SSTD = 11 (PCM standard is used) It is not used in SPI mode.*
[6]	R	Reserved, forced at 0 by hardware
[5:4]	RW	**I2SSTD[1:0]:** I2S standard selection 00: I2S Philips standard. 01: MSB justified standard (left justified) 10: LSB justified standard (right justified) 11: PCM standard **Note:** *For correct operation, these bits should be configured when the I2S is disabled.*
[3]	RW	**CKPOL:** Steady state clock polarity 0: I2S clock steady state is low level 1: I2S clock steady state is high level **Note:** *For correct operation, this bit should be configured when the I2S is disabled.* *This bit is not used in SPI mode*
[2:1]	RW	**DATLEN[1:0]:** 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, these bits should be configured when the I2S is disabled.* *This bit is not used in SPI mode.*
[0]	RW	**CKPOL:** Channel length (number of bits per audio channel) 0: 16-bit wide 1: 32-bit wide The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in. Not used in SPI mode. **Note:** *For correct operation, this bit should be configured when the I2S is disabled.*

21.5.9 SPI_I2S prescaler register(SPI_I2SPR)

Register	Address offset	Access	Reset value	Description
SPI_I2SPR	0x20	R	0x0000_0002	SPI_I2S prescaler register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						MCKOE	ODD
7	6	5	4	3	2	1	0
I2SDIV[7:0]							

SPI_I2S prescaler register(SPI_I2SPR)bit description

Bit	Access	Description

[31:10]	R	Reserved, must be kept at reset value
[9]	R	**MCKOE:** Master clock output enable 0: Master clock output is disabled 1: Master clock output is enabled **Note:** *This bit should be configured when the I2S is disabled. It is used only when the I2S is in master mode* *This bit is not used in SPI mode*
[8]	R	**ODD:** Odd factor for the prescaler 0: real divider value is = I2SDIV *2 1: real divider value is = (I2SDIV * 2)+1 **Note:** *This bit should be configured when the I2S is disabled. It is used only when the I2S is in master mode* *Refer to section 21.4.2 , Not used in SPI mode.*
[7:0]	R	**I2SDIV[7:0]:** I2S Linear prescaler I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values. **Note:** *These bits should be configured when the I2S is disabled. It is used only when the I2S is in master mode* *Refer to section 21.4.2 , Not used in SPI mode.*

22 USART

22.1 USART introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a fractional baud rate generator.

It supports synchronous one-way communication and half-duplex single wire communication. It also supports the LIN (local interconnection network), Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). It allows multiprocessor communication.

High speed data communication is possible by using the DMA for multibuffer configuration.

22.2 USART main features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- Fractional baud rate generator systems
  - A common programmable transmit and receive baud rates up to 4.5 MBits/s
- Programmable data word length (8 or 9 bits)
- Configurable stop bits - support for 1 or 2 stop bits
- LIN Master Synchronous Break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- Transmitter clock output for synchronous transmission
- IrDA SIR Encoder Decoder
  - Support for 3/16 bit duration for normal mode
- Smartcard Emulation Capability
  - The Smartcard interface supports the asynchronous protocol Smartcards as defined in ISO 7816-3 standards
  - 0.5, 1.5 Stop Bits for Smartcard operation
- Single wire half duplex communication
- Configurable multibuffer communication using DMA (direct memory access)
  - Buffering of received/transmitted bytes in reserved SRAM using centralized DMA
- Separate enable bits for Transmitter and Receiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - End of Transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte

- Four error detection flags:
  - Overrun error
  - Noise error
  - Frame error
  - Parity error
- Ten interrupt sources with flags:
  - CTS changes
  - LIN break detection
  - Transmit data register empty
  - Transmission complete
  - Receive data register full
  - Idle line received
  - Overrun error
  - Framing error
  - Noise error
  - Parity error
- Multiprocessor communication - enter into mute mode if address match does not occur
- Wake up from mute mode (by idle line detection or address mark detection)
- Two receiver wakeup modes: Address bit (MSB, 9 th bit), Idle line

22.3 USART functional description

The interface is externally connected to another device by three pins (see Figure 22.3-1). Any USART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

RX: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

TX: Transmit Data Output. When the transmitter is disabled, the output pin returns to its IO port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire and smartcard modes, this IO is used to transmit and receive the data (at USART level, data are then received on SW_RX).

Through these pins, serial data is transmitted and received in normal USART mode as frames comprising:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (8 or 9 bits) least significant bit first
- 0.5,1, 1.5, 2 Stop bits indicating that the frame is complete
- This interface uses a fractional baud rate generator - with a 12-bit mantissa and 4-bit fraction
- A status register (USART_SR)
- Data register (USART_DR)
- A baud rate register (USART_BRR) - 12-bit mantissa and 4-bit fraction.

- A Guardtime Register (USART_GTPR) in case of Smartcard mode.

Refer to Section 22.6: USART registers for the definition of each bit.

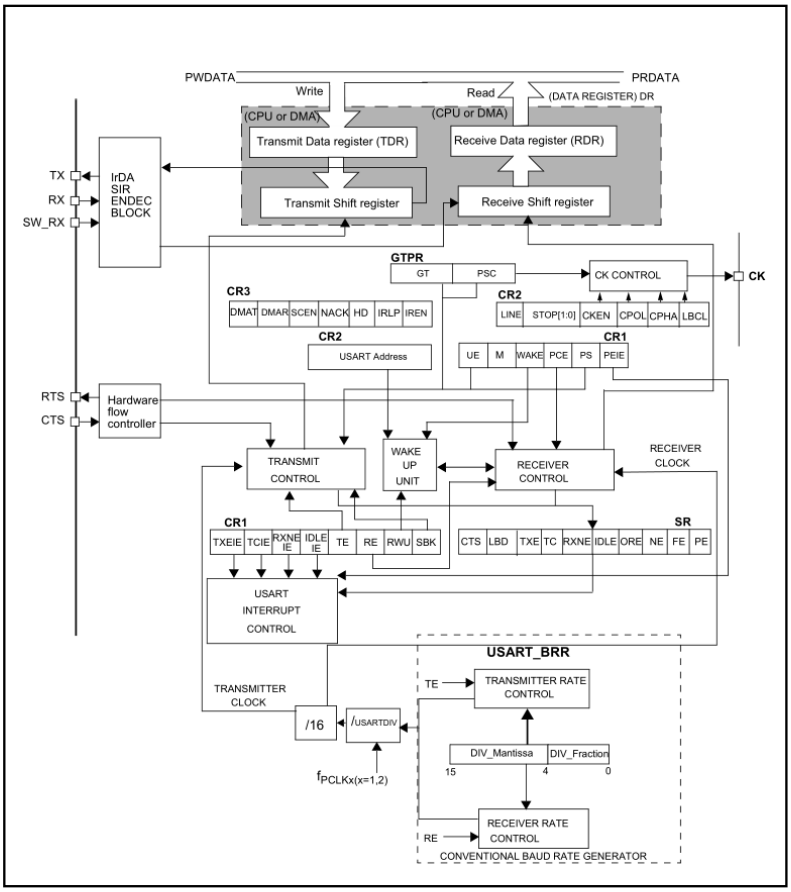
The following pin is required to interface in synchronous mode:

- **CK:** Transmitter clock output. This pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel data can be received synchronously on RX. This can be used to control peripherals that have shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable. In Smartcard mode, CK can provide the clock to the smartcard.

The following pins are required in Hardware flow control mode:

- **CTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **RTS:** Request to send indicates that the USART is ready to receive a data (when low).

Fig 22.3-1 USART block diagram



22.3.1 USART character description

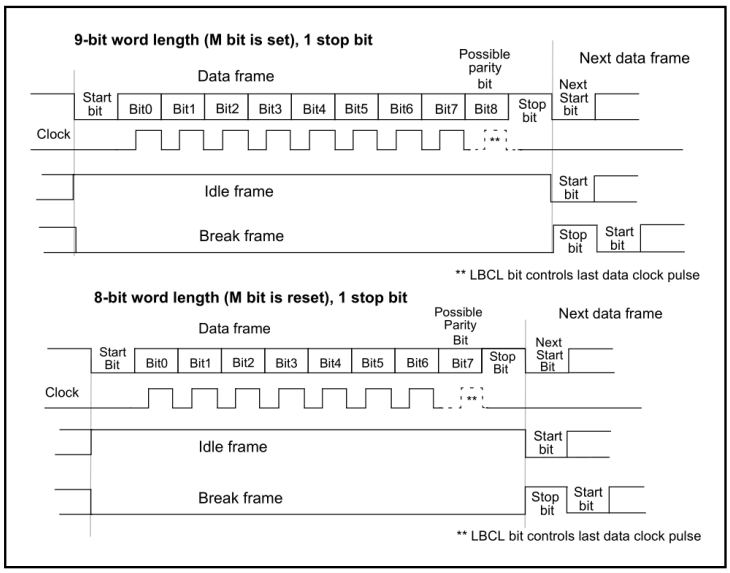
Word length may be selected as being either 8 or 9 bits by programming the M bit in the USART_CR1 register (see Figure 22.3-2).

The TX pin is in low state during the start bit. It is in high state during the stop bit. An Idle character

is interpreted as an entire frame of “1”s followed by the start bit of the next frame which contains data (The number of “1”‘s will include the number of stop bits). A Break character is interpreted on receiving “0”s for a frame period. At the end of the break frame the transmitter inserts either 1 or 2 stop bits (logic “1” bit) to acknowledge the start bit.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver. The details of each block is given below.

Fig 22.3-2 Word length programming



22.3.2 Transmitter

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. 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 pulses are output on the CK pin.

Character transmission

During a USART transmission, data shifts out least significant bit first on the TX pin. In this mode, the USART_DR register consists of a buffer (TDR) between the internal bus and the transmit shift register(see Figure 22.3-1)。

Every character is preceded by a start bit, which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by USART: 0.5, 1, 1.5 and 2 stop bits.

Note:

1. *The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters will get frozen. The current data being transmitted will be lost.*
2. *An idle frame will be sent after the TE bit is enabled.*

Configurable stop bits

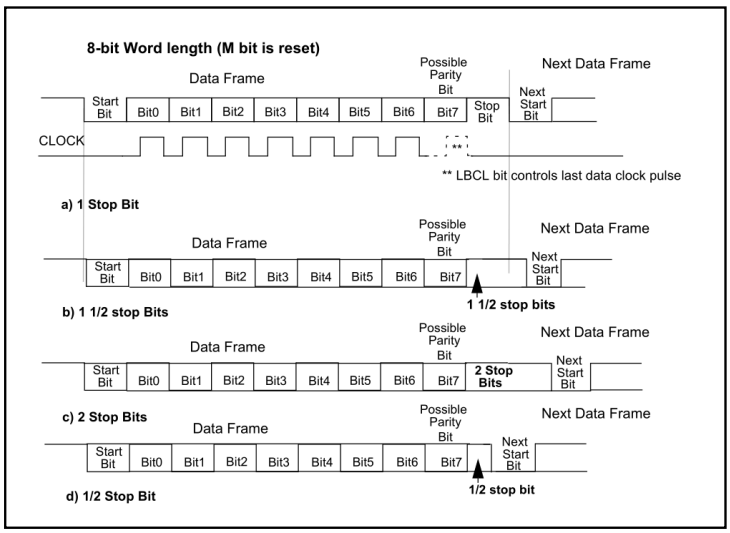
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

1. 1 stop bit: This is the default value.
2. 2 stop bits: This is supported by normal USART, single-wire and modem modes.
3. 0.5 stop bit: To be used when receiving data in Smartcard mode.
4. 1.5 stop bits: To be used when transmitting and receiving data in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits followed by the configured number of stop bits (when m = 0) and 11 low bits followed by the configured number of stop bits (when m = 1). It is not possible to transmit long breaks (break of length greater than 10/11 low bits).

Fig 22.3-3 Configurable stop bits



Procedure:

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
2. Program the M bit in USART_CR1 to define the word length.
3. Program the number of stop bits in USART_CR2.
4. Select DMA enable (DMAT) in USART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
5. Select the desired baud rate using the USART_BRR register.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_DR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.
8. After writing the last data into the USART_DR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.

Single byte communication

The TXE bit is always cleared by a write to the data register. The TXE bit is set by hardware and it indicates:

- The data has been moved from TDR to the shift register and the data transmission has started.
- The TDR register is empty.
- The next data can be written in the USART_DR register without overwriting the previous data

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is taking place, a write instruction to the USART_DR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the USART_DR register places the data directly in the shift register, the data transmission starts, and the TXE bit is immediately set.

If a frame is transmitted (after the stop bit) and the TXE bit is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

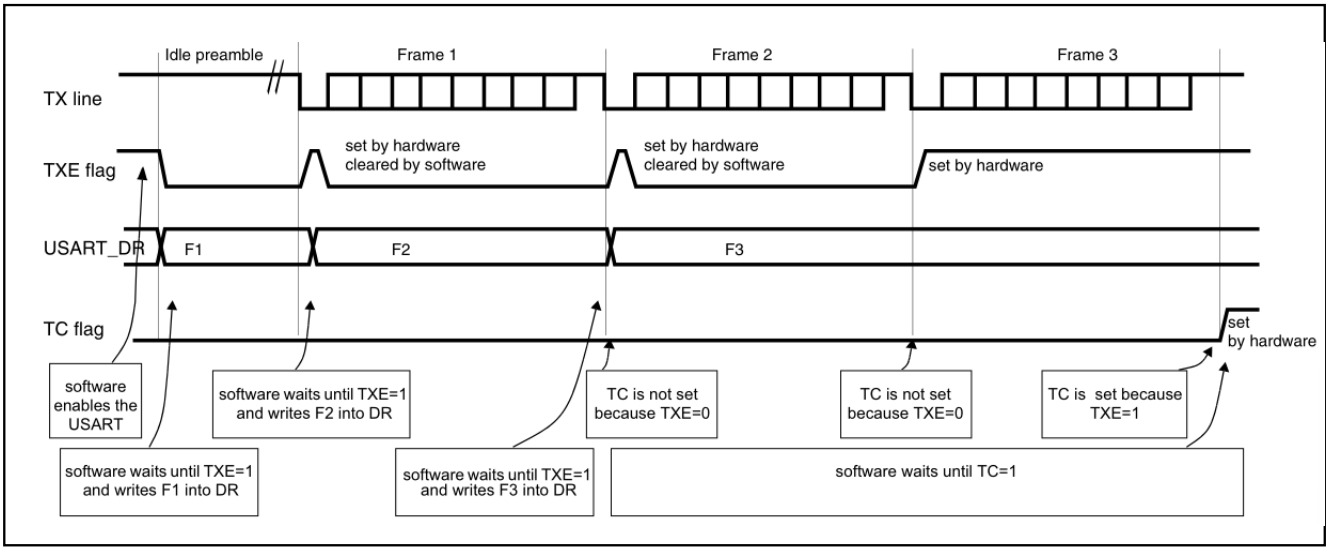
After writing the last data into the USART_DR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low-power modeFigure 22.3-4

The TC bit is cleared by the following software sequence:

- A read from the USART_SR register
- A write to the USART_DR register

Note: *The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for Multibuffer communication.*

Fig 22.3-4 TC/TXE behavior when transmitting



Break characters

Setting the SBK bit transmits a break character. The break frame length depends on the M bit (Figure 22.3-1)

If the SBK bit is set to '1' a break character is sent on the TX line after completing the current character transmission. This bit is reset by hardware when the break character is completed (during the stop bit of the break character). The USART inserts a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

Note: *If the software resets the SBK bit before the commencement of break transmission, the break character will not be transmitted. For two consecutive breaks, the SBK bit should be set after the stop bit of the previous break.*

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

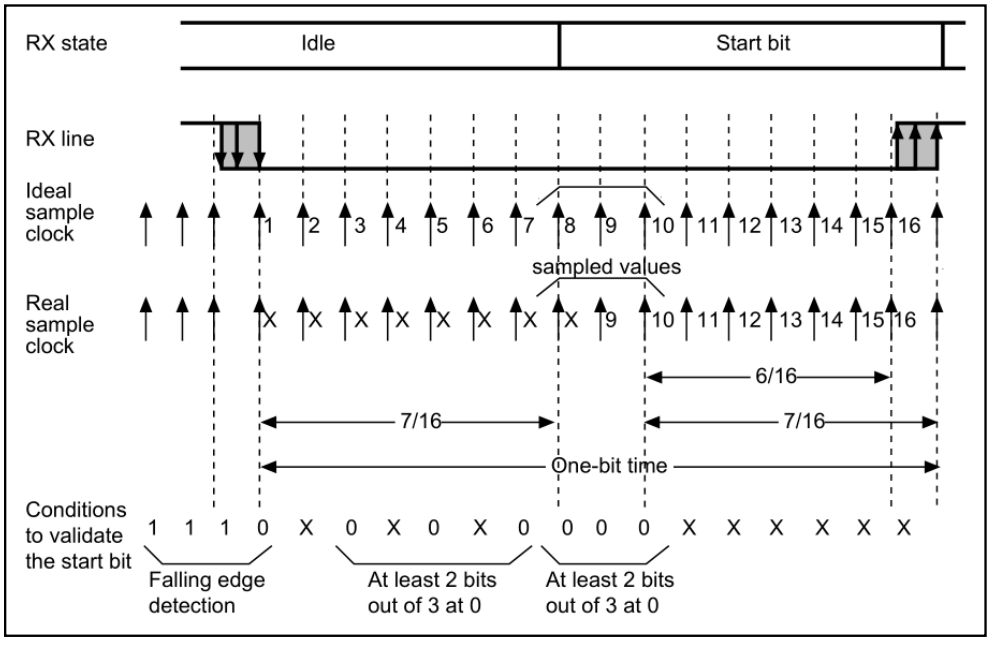
22.3.3 Receiver

The USART can receive data words of either 8 or 9 bits depending on the M bit in the USART_CR1 register.

Start bit detection

In the USART, the start bit is detected when a specific sequence of samples is recognized.
This sequence is: 1-1-1-0-X-0-X-0-X-0-0-0-0

Fig 22.3-5 Start bit detection



If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state

(no flag is set) where it waits for a falling edge.

The start bit is confirmed (RXNE flag set, interrupt generated if RXNEIE=1) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated (RXNE flag set, interrupt generated if RXNEIE=1) but the NE noise flag is set if, for both samplings, at least 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits). If this condition is not met, the start detection aborts and the receiver returns to the idle state (no flag is set).

If, for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0, the start bit is validated but the NE noise flag bit is set

Character reception

During a USART reception, data shifts in least significant bit first through the RX pin. In this mode, the USART_DR register consists of a buffer (RDR) between the internal bus and the received shift register.

Procedure:

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
2. Program the M bit in USART_CR1 to define the word length.
3. Program the number of stop bits in USART_CR2.
4. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication. STEP 3
5. Select the desired baud rate using the baud rate register USART_BRR
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- The RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- An interrupt is generated if the RXNEIE bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer, RXNE is set after every byte received and is cleared by the DMA read to the Data Register.
- In single buffer mode, clearing the RXNE bit is performed by a software read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

Note: *The RE bit should not be reset while receiving data. If the RE bit is disabled during reception, the reception of the current byte will be aborted.*

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as a data received character plus an interrupt if the IDLEIE bit is set.

Overrun error

An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared.

The RXNE flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- The ORE bit is set.
- The RDR content will not be lost. The previous data is available when a read to USART_DR is performed.
- The shift register will be overwritten. After that point, any data received during overrun is lost.
- An interrupt is generated if either the RXNEIE bit is set or both the EIE and DMAR bits are set.
- The ORE bit is reset by a read to the USART_SR register followed by a USART_DR register read operation.

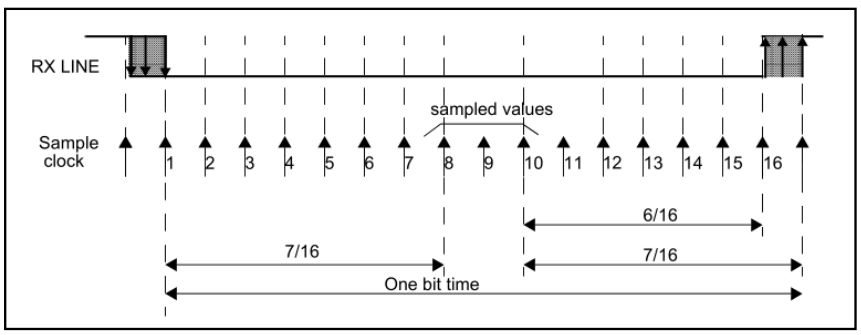
The ORE bit, when set, indicates that at least 1 data has been lost. There are two possibilities:

1. if RXNE=1, then the last valid data is stored in the receive register RDR and can be read,
2. if RXNE=0, then it means that the last valid data has already been read and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received. It may also occur when the new data is received during the reading sequence (between the USART_SR register read access and the USART_DR read access).

Noise error

Over-sampling techniques are used (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise.

Fig 22.3-6 Data sampling for noise detection



Tab 22.3-1 Noise detection from sampled data

Sampled value	NE status	Received bit value	Data validity
000	0	0	Valid
001	1	0	Not Valid
010	1	0	Not Valid
011	1	1	Not Valid
100	1	0	Not Valid
101	1	1	Not Valid
110	1	1	Not Valid
111	0	1	Valid

When noise is detected in a frame:

- The NE is set at the rising edge of the RXNE bit.
- The invalid data is transferred from the Shift register to the USART_DR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register。

The NE bit is reset by a USART_SR register read operation followed by a USART_DR register read operation.

Framing error

A framing error is detected when:

The stop bit is not recognized on reception at the expected time, following either a de- synchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware
- The invalid data is transferred from the Shift register to the USART_DR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by a USART_SR register read operation followed by a USART_DR register read operation

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

1. **0.5 stop bit (reception in Smartcard mode):** No sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.

2. **1 stop bit:** Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
3. **1.5 stop bits (Smartcard mode):** When transmitting in Smartcard mode, the device must check that the data is correctly sent. Thus the receiver block must be enabled (RE =1 in the USART_CR1 register) and the stop bit is checked to test if the smartcard has detected a parity error. In the event of a parity error, the smartcard forces the data signal low during the sampling - NACK signal-, which is flagged as a framing error. Then, the FE flag is set with the RXNE at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be decomposed into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through. Refer to Section 27.3.11 for more details.
4. **2 stop bits:** Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. If a framing error is detected during the first stop bit the framing error flag will be set. The second stop bit is not checked for framing error. The RXNE flag will be set at the end of the first stop bit.

22.3.4 Fractional baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the Mantissa and Fraction values of USARTDIV

$$Tx/BaudRate = \frac{f_{CK}}{16 * USARTDIV} \tag{22.1}$$

f_{CK} : Input clock to the peripheral (PCLK1 for USART2, 3, 4, 5 or PCLK2 for USART1)

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

Note: *The baud counters are updated with the new value of the Baud registers after a write to USART_BRR. Hence the Baud rate register value should not be changed during communication.*

How to derive USARTDIV from USART_BRR register values
Example 1:

If DIV_Mantissa = 0d27 and DIV_Fraction = 0d12 (USART_BRR = 0x1BC), then
Mantissa (USARTDIV) = 0d27
Fraction (USARTDIV) = 12/16 = 0d0.75
Therefore USARTDIV = 0d27.75

Example 2:

To program USARTDIV = 0d25.62
This leads to:
DIV_Fraction = 16*0d0.62 = 0d9.92
The nearest real number is 0d10 = 0xA
DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19
Then, USART_BRR = 0x19A hence USARTDIV = 0d25.625

Example 3:

To program USARTDIV = 0d50.99
This leads to:
DIV_Fraction = 16*0d0.99 = 0d15.84
The nearest real number is 0d16 = 0x10 => overflow of DIV_frac[3:0] => carry must be added up to the mantissa
DIV_Mantissa = mantissa (0d50.990 + carry) = 0d51 = 0x33
Then, USART_BRR = 0x330 hence USARTDIV = 0d51.000

Tab 22.3-2 Error calculation for programmed baud rates

Baud rate		fPCLK = 48MHz			fPCLK = 96MHz		
S.No	Kbps	Actual	Value programmed in the Baud Rate register	Error%	Actual	Value programmed in the Baud Rate register	Error%
1	2.4	2.400	1250	0%	2.400	2500	0%
2	9.6	9.600	312.5	0%	9.600	625	0%
3	19.2	19.2	156.25	0%	19.200	312.5	0%
4	57.6	57.623	52.0625	0.04%	57.623	104.125	0.04%
5	115.2	115.385	26	0.16%	115.246	52.0625	0.04%
6	230.4	230.769	13	0.16%	230.769	26	0.16%
7	468.8	470.588	6.375	0.38%	470.588	12.75	0.38%
8	921.6	923.077	3.25	0.16%	923.077	6.5	0.16%
9	2250	2285.714	1.3125	1.59%	2285.714	2.625	1.59%
10	4500	NA	NA	NA	4571.428	1.3125	1.59%

Note: *Error = Defined as (Calculated Baud Rate - Desired Baud Rate) / Desired Baud Rate.*

The lower the CPU clock the lower will be the accuracy for a particular Baud rate. The upper limit of the achievable baud rate can be fixed with this data.

Only USART1 is clocked with PCLK2 (96 MHz max). Other USARTs are clocked with PCLK1 (48 MHz max).

22.3.5 USART receiver’s tolerance to clock deviation

The USART’s asynchronous receiver works correctly only if the total clock system deviation is smaller than the USART receiver tolerance. The causes that contribute to the total deviation are:

- DTRA: Deviation due to the transmitter error (which also includes the deviation of the transmitter local oscillator)
- DQUANT: Error due to the baud rate quantization of the receiver
- DREC: Deviation of the receiver’s local oscillator
- DTCL: Deviation due to the transmission line (generally due to the transceivers that can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

DTRA + DQUANT + DREC + DTCL < USART receiver tolerance
The USART receiver tolerance to properly receive data is equal to the maximum tolerated deviation and depends on the following choices:

- 10- or 11-bit character length defined by the M bit in the USART_CR1 register
- use of fractional baud rate or not

Tab 22.3-3 USART receiver tolerance when DIV_Fraction is 0

M bit	NF is an error	NF is don’t care
0	3.75%	4.375%
1	3.41%	3.97%

Tab 22.3-4 USART receiver tolerance when DIV_Fraction is different from 0

M bit	NF is an error	NF is don’t care
0	3.33%	3.88%
1	3.03%	3.53%

Note1: *The figures specified in Table 22.3-3 and Table 22.3-4 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M=0 (11-bit times when M=1).*

Note2: *The transmitter should insert at least 1 bit idle cycle time between two continues data transfer to avoid receive data error due to clock deviation*

22.3.6 Multiprocessor communication

There is a possibility of performing multiprocessor communication with the USART (several USARTs connected in a network). For instance, one of the USARTs can be the master, its TX output is connected to the RX input of the other USART. The others are slaves, their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART_CR1 register is set to 1. RWU can be controlled automatically by hardware or written by the software under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

1. Idle Line detection if the WAKE bit is reset,

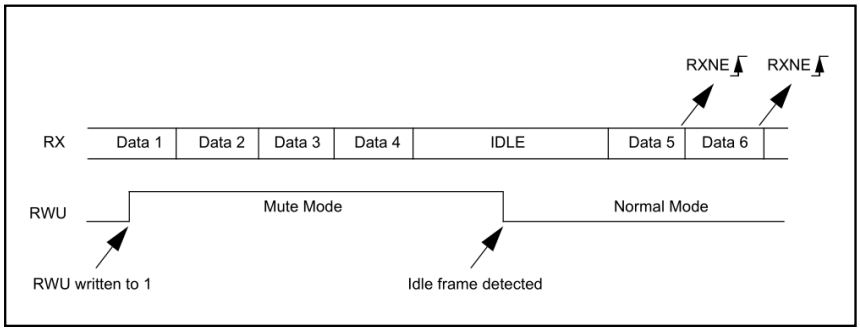
2. Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The USART enters mute mode when the RWU bit is written to 1. It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART_SR register. RWU can also be written to 0 by software.

An example of mute mode behavior using idle line detection is given 22.3-7.

Fig 22.3-7 Mute mode using Idle line detection



Address mark detection (WAKE=1)

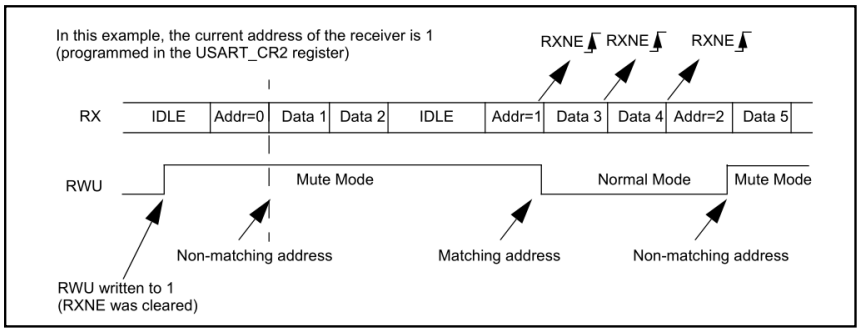
In this mode, bytes are recognized as addresses if their MSB is a '1' else they are considered as data. In an address byte, the address of the targeted receiver is put on the 4 LSB. This 4-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt nor DMA request is issued as the USART would have entered mute mode.

It exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE bit is set for the address character since the RWU bit has been cleared.

The RWU bit can be written to as 0 or 1 when the receiver buffer contains no data (RXNE=0 in the USART_SR register). Otherwise the write attempt is ignored. An example of mute mode behavior using address mark detection is given in 22.3-8

Fig 22.3-8 Mute mode using address mark detection



22.3.7 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bit, the possible USART frame formats are as listed in Table 22.3-5

Tab 22.3-5 Frame formats

M bit	PCE bit	USART frame
0	0	Start Bit - 8 bit data - Stop Bit
0	1	Start Bit - 7 bit data - Parity Bit - Stop Bit
1	0	Start Bit - 9 bit data - Stop Bit
1	1	Start Bit - 8 bit data - Parity Bit - Stop Bit

Note: *In case of wake up by an address mark, the MSB bit of the data is taken into account and not the parity bit*

- **Even parity**: the parity bit is calculated to obtain an even number of "1s" inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.
  Ex: data=00110101; 4 bits set => parity bit will be 0 if even parity is selected (PS bit in USART_CR1 = 0)
- **Odd parity**: the parity bit is calculated to obtain an odd number of "1s" inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.
  Ex: data=00110101; 4 bits set => parity bit will be 1 if odd parity is selected (PS bit in USART_CR1 = 1).
- **Transmission mode**: If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of "1s" if even parity is selected (PS=0) or an odd number of "1s" if odd parity is selected (PS=1)). If the parity check fails, the PE flag is set in the USART_SR register and an interrupt is generated if PEIE is set in the USART_CR1 register。

22.3.8 LIN (local interconnection network) mode

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0], CLKEN in the USART_CR2 register
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The same procedure explained in Section 22.3.2 has to be applied for LIN Master transmission than for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBK bit sends 13‘0’bits as a break character. Then a bit of value ‘1’ is sent to allow the next start detection.

LIN reception

A break detection circuit is implemented in the USART. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as ‘0’, and are followed by a delimiter character, the LBD flag is set in USART_SR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1’ is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0’, which will be the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1’, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the Figure 22.3-9. Examples of break frames are given on Figure 22.3-10

Fig 22.3-9 Break detection in LIN mode (11-bit break length - LBDL bit is set)

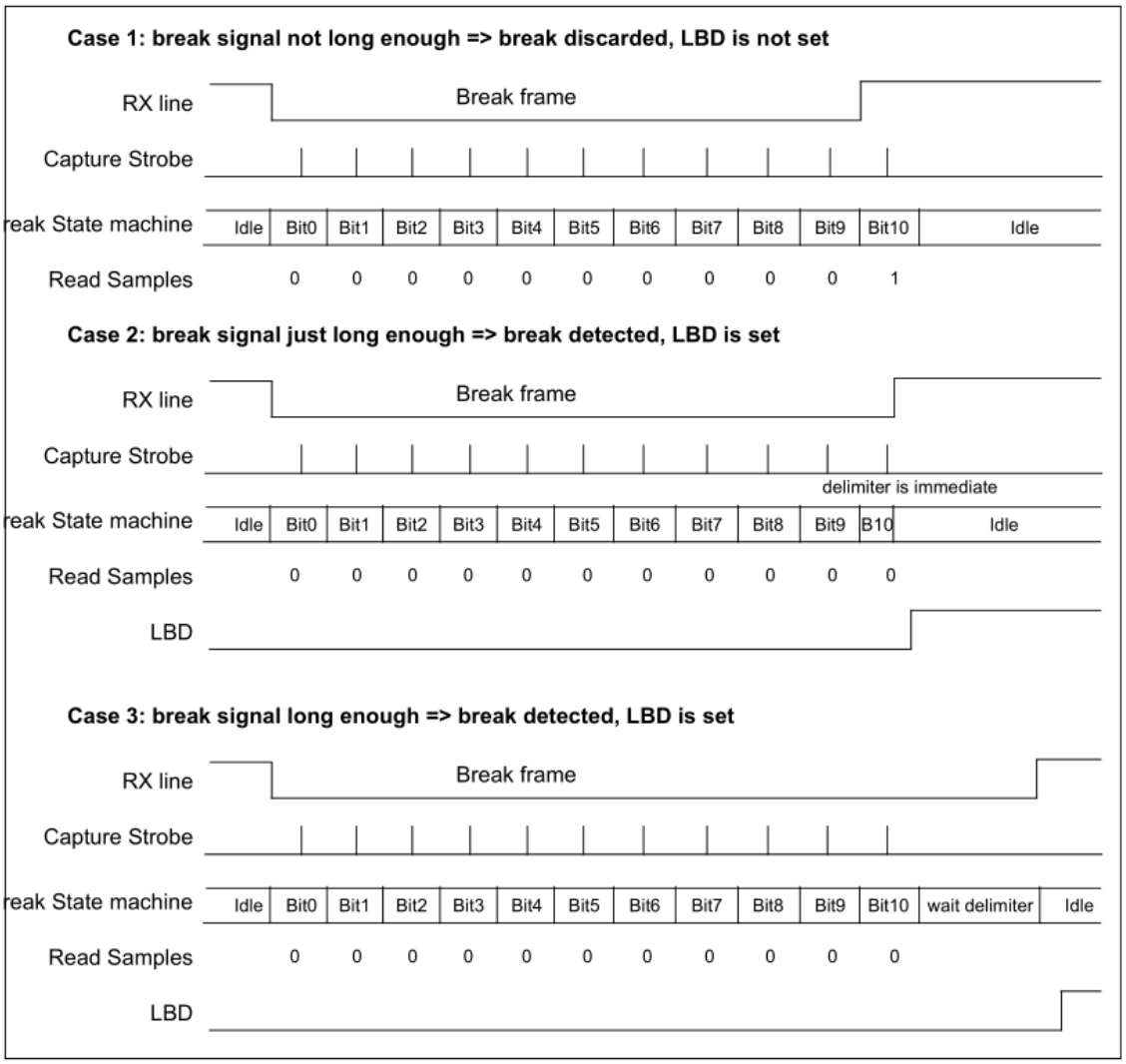
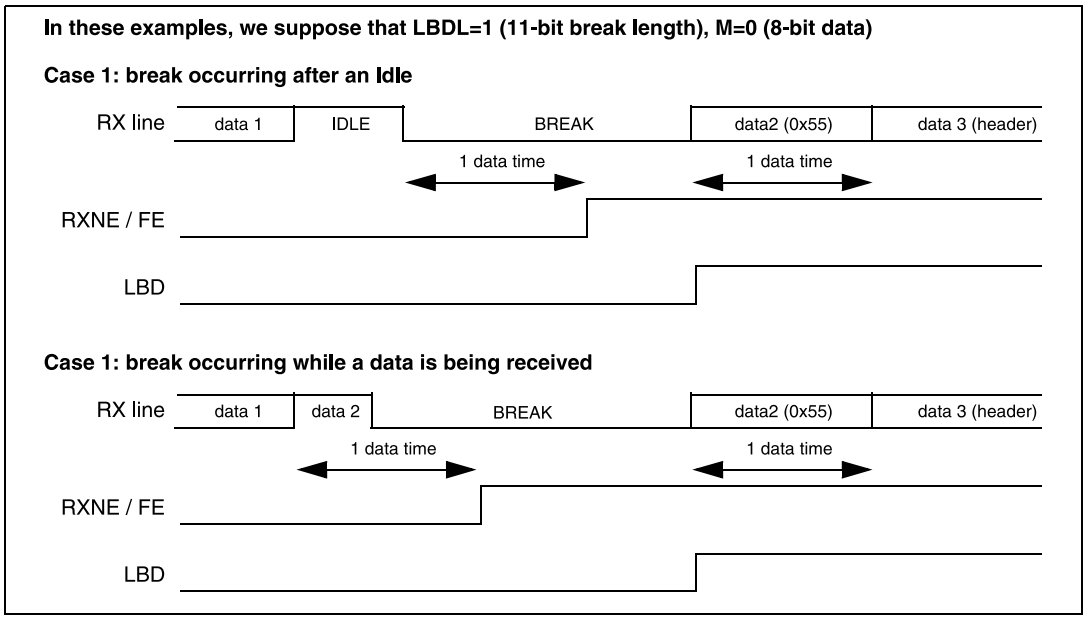


Fig 22.3-10 Break detection in LIN mode vs. Framing error detection



22.3.9 USART synchronous mode

The synchronous mode is selected by writing the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register

The USART allows the user to control a bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register clock pulses will or will not be generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register allows the user to select the clock polarity, and the CPHA bit in the USART_CR2 register allows the user to select the phase of the external clock (see Figure 22.3-11, Figure 22.3-12 and Figure 22.3-13).

During idle, preamble and send break, the external CK clock is not activated.

In synchronous mode the USART transmitter works exactly like in asynchronous mode. But as CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In this mode the USART receiver works in a different manner compared to the asynchronous mode. If RE=1, the data is sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note:

1. *The CK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and a data is being transmitted (the data register USART_DR has been written). This means that it is not possible to receive a synchronous data without transmitting data.*
2. *The LBCL, CPOL and CPHA bits have to be selected when both the transmitter and the receiver are disabled (TE=RE=0) to ensure that the clock pulses function correctly. These bits should not be changed while the transmitter or the receiver is enabled.*
3. *It is advised that TE and RE are set in the same instruction to minimize the setup and the hold time of the receiver.*
4. *The USART supports master mode only: it cannot receive or send data related to an input clock (CK is always an output).*

Fig 22.3-11 USART example of synchronous transmission

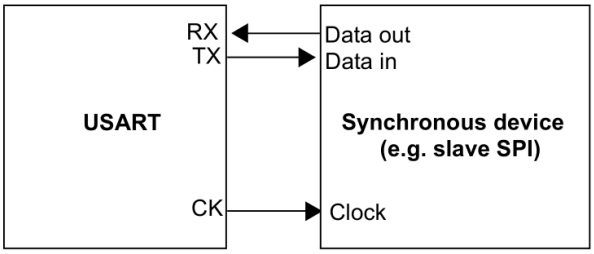


Fig 22.3-12 USART data clock timing diagram (M=0)

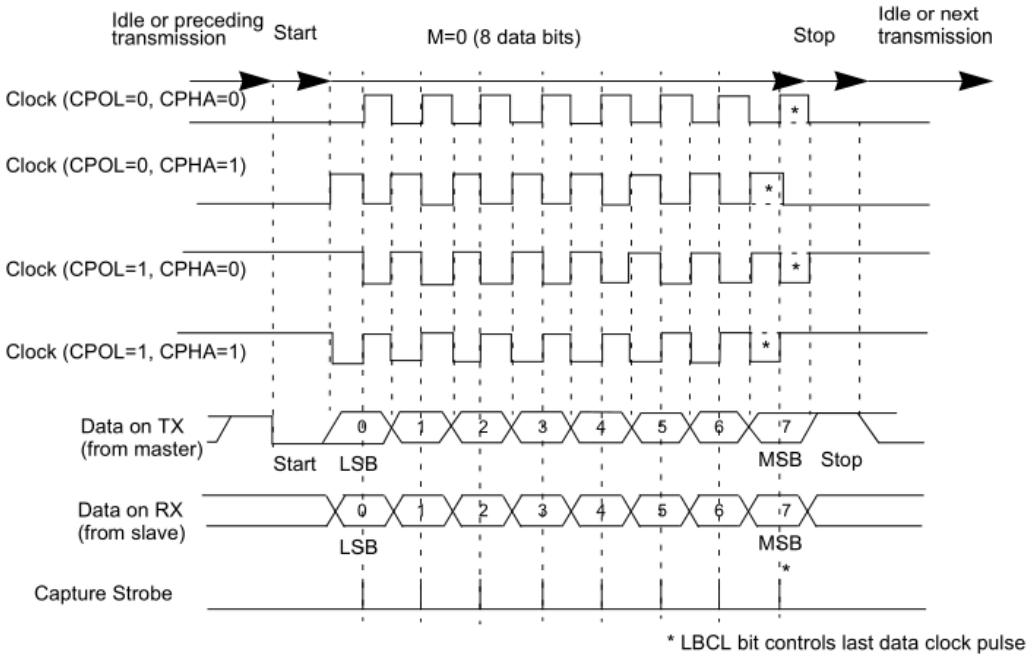


Fig 22.3-13 USART data clock timing diagram (M=1)

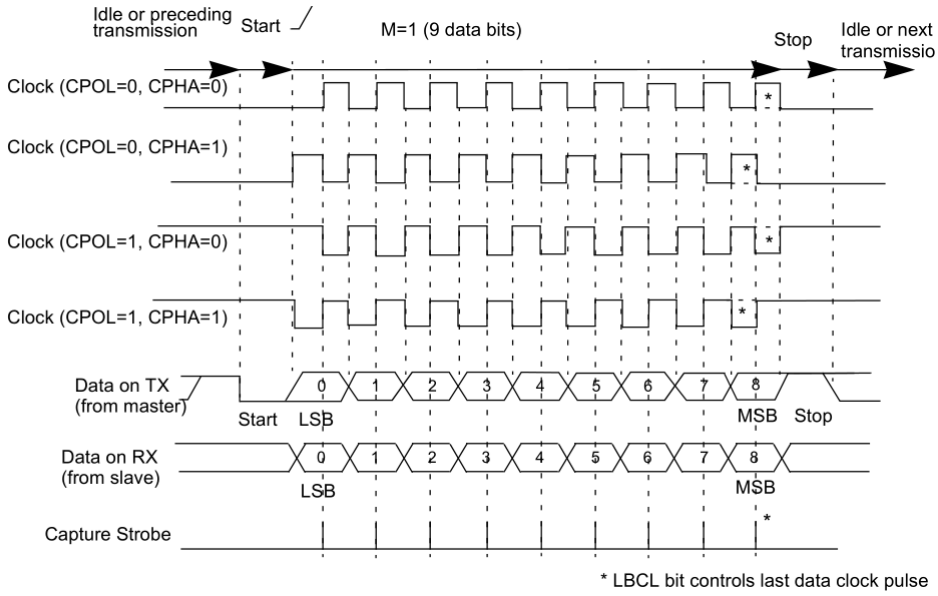
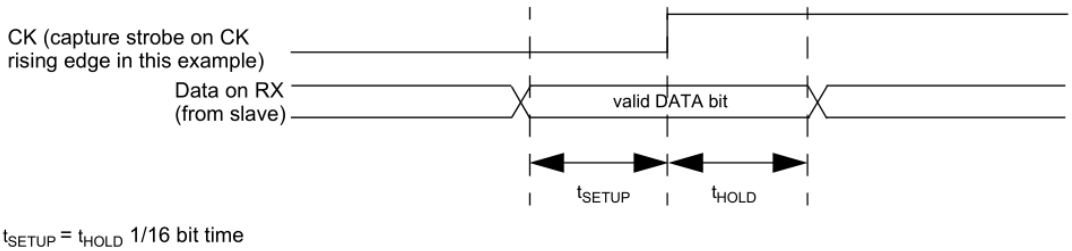


Fig 22.3-14 RX data setup/hold time



$t_{SETUP} = t_{HOLD}$ 1/16 bit time

Note: *The function of CK is different in Smartcard mode. Refer to the Smartcard mode section for more details.*

22.3.10 Single-wire half-duplex communication

The single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in 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 connected internally. The selection between half- and full-duplex communication is made with a control bit ‘HALF DUPLEX SEL’ (HDSEL in USART_CR3).

As soon as HDSEL is written to 1:

- RX is no longer used,
- TX is always released when no data is transmitted. Thus, it acts as a standard IO in idle or in reception. It means that the IO must be configured so that TX is configured as floating input (or output high open-drain) when not driven by the USART.

Apart from this, the communications are similar to what is done in normal USART mode. The conflicts on the line must be managed by the software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continue to occur as soon as a data is written in the data register while the TE bit is set.

22.3.11 Smartcard

The Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

Moreover, the CLKEN bit may be set in order to provide a clock to the smartcard.

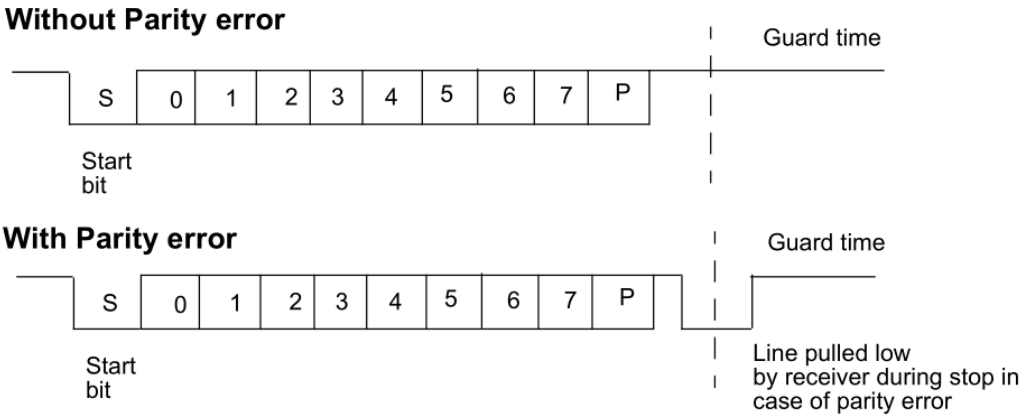
The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO 7816-3 standard. The USART should be configured as:

- 8 bits plus parity: where M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving : where STOP=’11’ in the USART_CR2 register.

Note: *It is also possible to choose 0.5 stop bit for receiving but it is recommended to use 1.5 stop bits for both transmitting and receiving to avoid switching between the two configurations.*

Figure 22.3-15 shows examples of what can be seen on the data line with and without parity error.

Fig 22.3-15 ISO 7816-3 asynchronous protocol



When connected to a smartcard, the USART TX output drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard is a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register will start shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- If a parity error is detected during reception of a frame programmed with a 0.5 or 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to USART has not been correctly received. This NACK signal (pulling transmit line low for 1 baud clock) will cause a framing error on the transmitter side (configured with 1.5 stop bits). The application can handle re-sending of data according to the protocol. A parity error is ‘NACK’ed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted.
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK will not be detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver will not detect the NACK as a start bit.

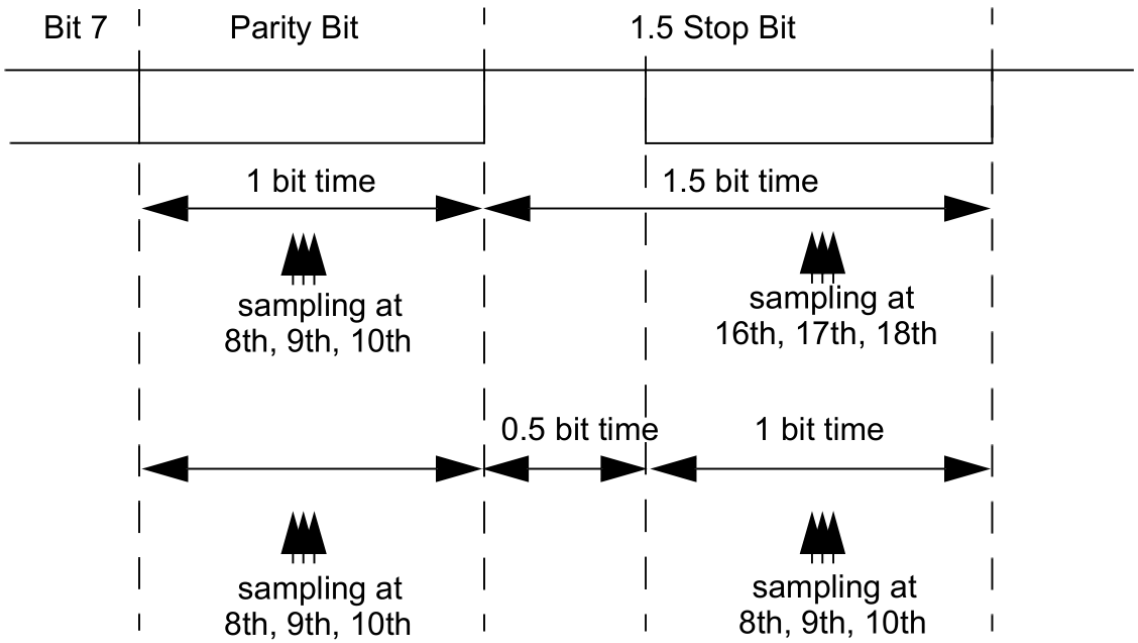
Note:

1. *A break character is not significant in Smartcard mode. A 0x00 data with a framing error will be treated as data and not as a break.*

2. *No IDLE frame is transmitted when toggling the TE bit. The IDLE frame (as defined for the other configurations) is not defined by the ISO protocol.*

Figure 22.3-16 details how the NACK signal is sampled by the USART. In this example, the USART transmits a data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Fig 22.3-16 Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In Smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USART_GTPR. CK frequency can be programmed from f CK /2 to f CK /62, where f CK is the peripheral input clock.

22.3.12 IrDA SIR ENDEC block

The IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see Figure 22.3-17).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and

outputs the received NRZ serial bit stream to USART. The decoder input is normally HIGH (marking state) in the idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (i.e. the USARTsends data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (USART receives decoded data from the USART), data on the TX from the USART to IrDA is not encoded by IrDA. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.
- A ‘0’ is transmitted as a high pulse and a ‘1’ is transmitted as a ‘0’. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see Figure 22.3-18).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 us. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the IrDA low-power Baud Register, USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOP bits in the USART_CR2 register must be configured to “1 stop bit”.

IrDA low-power mode

1.Transmitter

In low-power mode the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

2.Receiver

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection, the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in USART_GTPR).

Note:

1. *A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.*
2. *The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol)*

Fig 22.3-17 IrDA SIR ENDEC-block diagram

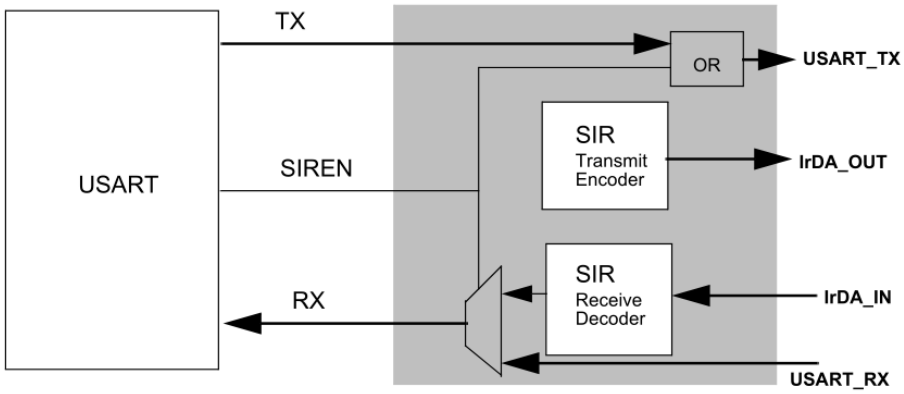
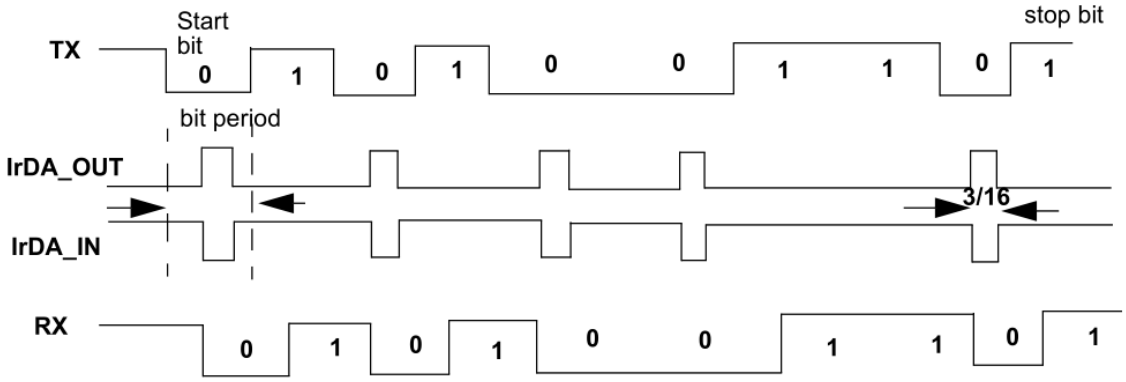


Fig 22.3-18 IrDA data modulation (3/16) -normal mode



22.3.13 Frequency splitting Settings for different baud rates

The accurate baud rate BRR values of serial ports at different frequencies need to be subtracted from the following difference values based on the calculated formula values

Baud rate	Serial Frequency (MHZ)	BRR difference
115200	96	3
115200	88	2
115200	80	3
115200	72	2
115200	64	2
115200	56	2
115200	48	1
115200	40	1
115200	32	1
115200	24	1
115200	16	0
115200	8	0

Baud rate	Serial Frequency (MHZ)	BRR difference
57600	96	7
57600	88	6
57600	80	5
57600	72	5
57600	64	5
57600	56	4
57600	48	3
57600	40	2
57600	32	2
57600	24	1
57600	16	1
57600	8	0

Baud rate	Serial Frequency (MHZ)	BRR difference
38400	96	10
38400	88	9
38400	80	8
38400	72	8
38400	64	7
38400	56	6
38400	48	6
38400	40	4
38400	32	3
38400	24	3
38400	16	1
38400	8	1

Baud rate	Serial Frequency (MHZ)	BRR difference
19200	96	23
19200	88	20
19200	80	18
19200	72	16
19200	64	15
19200	56	12
19200	48	11
19200	40	9
19200	32	7
19200	24	6
19200	16	3
19200	8	1

Baud rate	Serial Frequency (MHZ)	BRR difference
9600	96	45
9600	88	40
9600	80	35
9600	72	35
9600	64	30
9600	56	26
9600	48	21
9600	40	17
9600	32	15
9600	24	11
9600	16	7
9600	8	3

Baud rate	Serial Frequency (MHZ)	BRR difference
4800	96	83
4800	88	78
4800	80	71
4800	72	65
4800	64	59
4800	56	55
4800	48	48
4800	40	40
4800	32	30
4800	24	21
4800	16	15
4800	8	7

Baud rate	Serial Frequency (MHZ)	BRR difference
2400	96	165
2400	88	159
2400	80	154
2400	72	142
2400	64	123
2400	56	108
2400	48	93
2400	40	77
2400	32	62
2400	24	47
2400	16	31
2400	8	15

Baud rate	Serial Frequency (MHZ)	BRR difference
1200	96	0
1200	88	0
1200	80	0
1200	72	257
1200	64	227
1200	56	205
1200	48	185
1200	40	154
1200	32	123
1200	24	93
1200	16	62
1200	8	31

22.3.14 Continuous communication using DMA

The USART is capable of continuing communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: *User should refer to product specs for availability of the DMA controller. If DMA is not available in the product, you should use the USART as explained in Section 22.3.2 or 22.3.3 In the USART_SR register, user can clear the TXE/ RXNE flags to achieve continuous communication.*

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to the DMA specification) to the USART_DR register whenever the TXE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

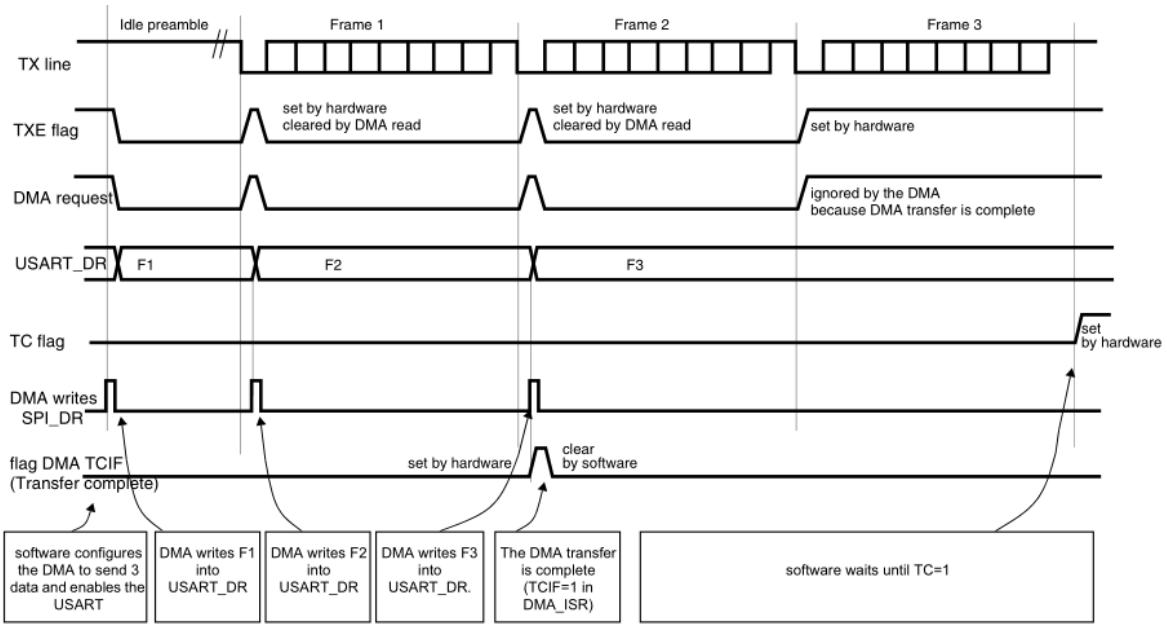
1. Write the USART_DR register address in the DMA control register to configure it as the destination of the transfer. The data will be moved to this address from memory after each TXE event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data will be loaded into the USART_DR register from this memory area after each TXE event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC bit in the SR register by writing 0 to it.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or entering the Stop mode. The software must wait until TC=1. The TC flag remains cleared during all

data transfers and it is set by hardware at the last frame’s end of transmission.

Fig 22.3-19 Transmission using DMA



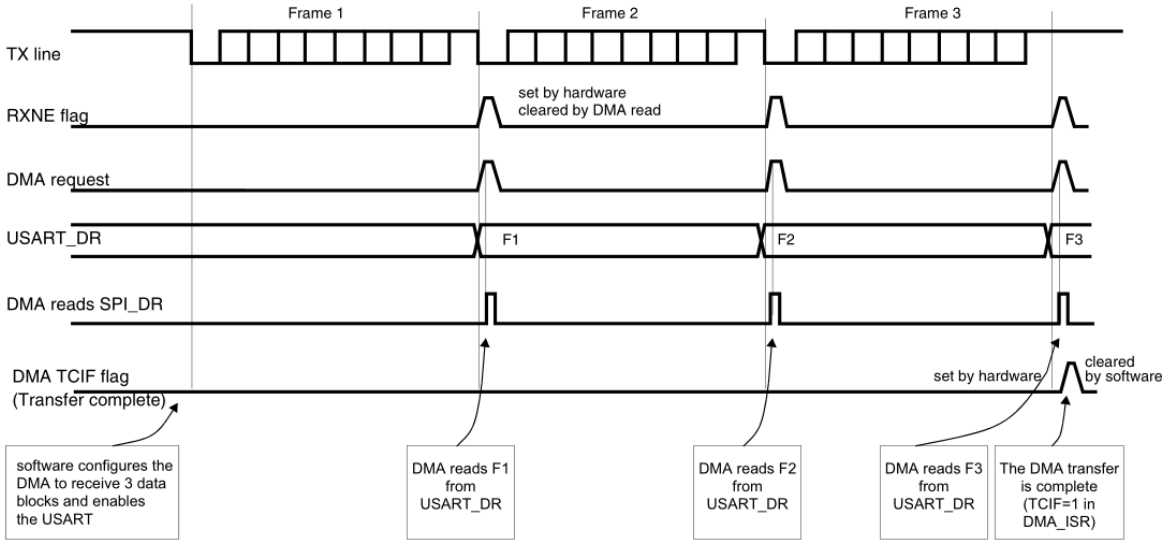
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data is loaded from the USART_DR register to a SRAM area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_DR register address in the DMA control register to configure it as the source of the transfer. The data will be moved from this address to the memory after each RXNE event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data will be loaded from USART_DR to this memory area after each RXNE event.
3. Configure the total number of bytes to be transferred in the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Fig 22.3-20 Reception using DMA



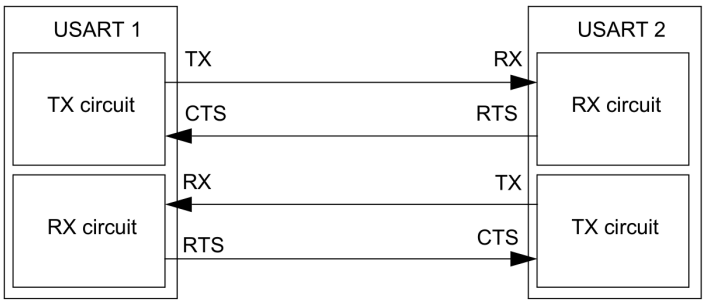
Error flagging and interrupt generation in multibuffer communication

In case of multibuffer communication if any error occurs during the transaction the error flag will be asserted after the current byte. An interrupt will be generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE in case of single byte reception, there will be separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which if set will issue an interrupt after the current byte with either of these errors.

22.3.15 Hardware flow control

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output.

Fig 22.3-21 Hardware flow control between two USARTs



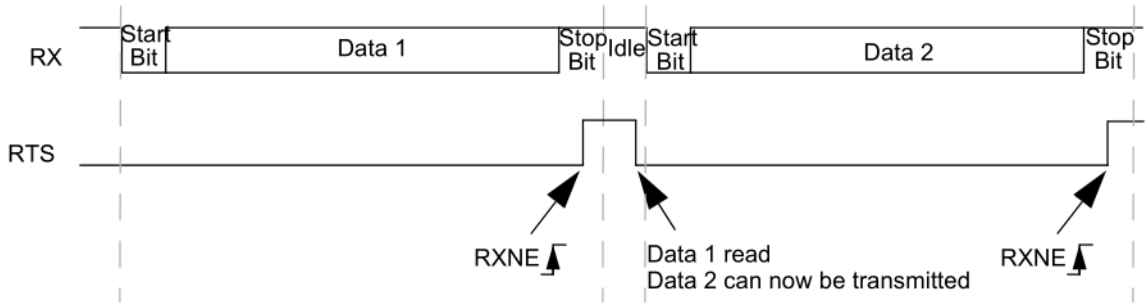
RTS and CTS flow control can be enabled independently by writing respectively RTSE and CTSE bits to 1 (in the USART_CR3 register).

RTS flow control

If the RTS flow control is enabled (RTSE=1), then RTS is asserted (tied low) as long as the USART receiver is ready to receive new data. When the receive register is full, RTS is deasserted, indicating that the

transmission is expected to stop at the end of the current frame.

Fig 22.3-22 RTS flow control

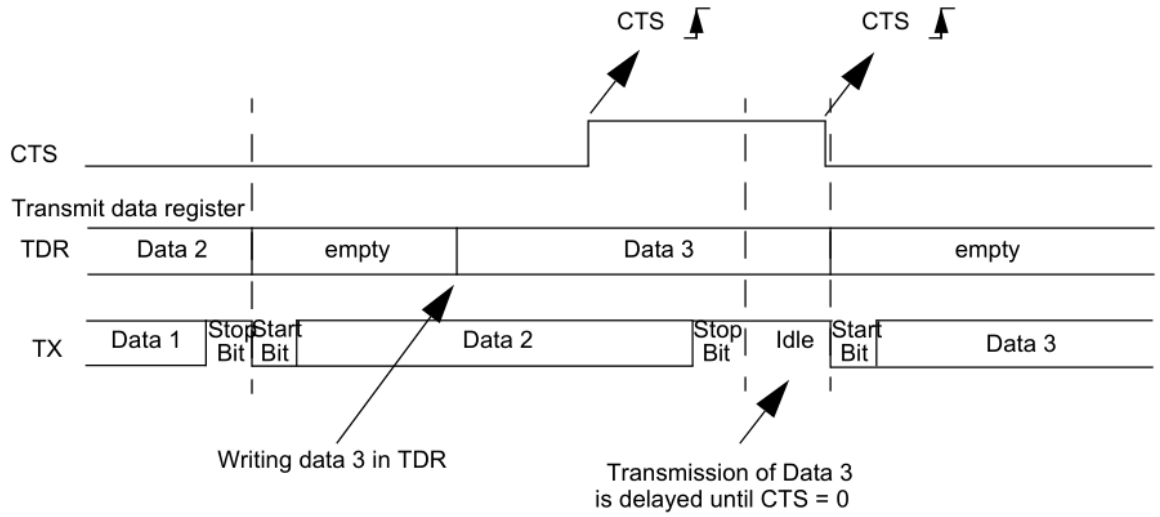


CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is asserted (tied low), then the next data is transmitted (assuming that a data is to be transmitted, in other words, if TXE=0), else the transmission does not occur. When CTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. The figure below shows an example of communication with CTS flow control enabled.

Fig 22.3-23 CTS flow control



22.4 USART interrupts

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TXE	TXEIE
CTS flag	CTS	CTSIE
Transmission complete	TC	TCIE
Received data ready to be read	TXNE	TXNEIE
Overrun error detected	ORE	
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
Break flag	LBD	LBDIE
Noise flag, Overrun error and Framing error in multibuffer communication	NE or ORE or FE	EIE(1)

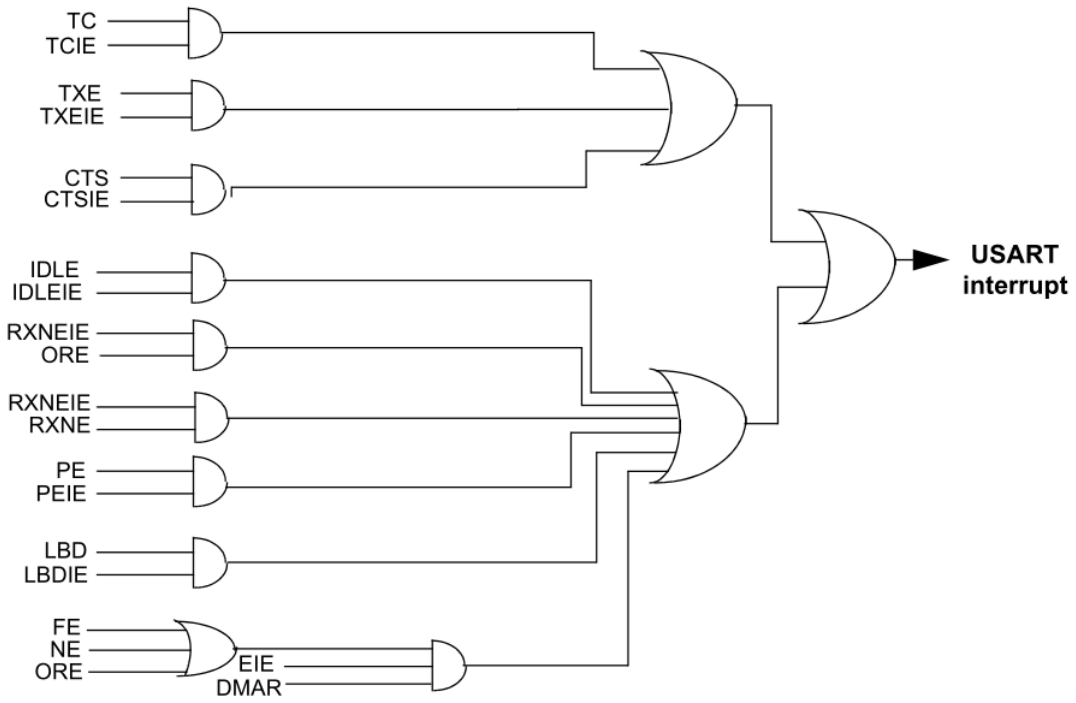
1:This bit is used only when data reception is performed by DMA

The USART interrupt events are connected to the same interrupt vector

- During transmission: Transmission Complete, Clear to Send or Transmit Data Register empty interrupt.
- While receiving: Idle Line detection, Overrun error, Receive Data register not empty, Parity error, LIN break detection, Noise Flag (only in multi buffer communication) and Framing Error (only in multi buffer communication).。

These events generate an interrupt if the corresponding Enable Control Bit is set.

Fig 22.4-1 USART interrupt mapping diagram



22.5 USART mode configuration

USART modes	USART1	USART2	USART3	UART4	UART5
Asynchronous mode	X	X	X	X	X
Hardware Flow Control	X	X	X	NA	NA
Multibuffer Communication (DMA)	X	X	X	X	NA
Multiprocessor Communication	X	X	X	X	X
Synchronous	X	X	X	NA	NA
Smartcard	X	X	X	NA	NA
Half-Duplex (Single-Wire mode)	X	X	X	X	X
IrDA	X	X	X	X	X
LIN	X	X	X	X	X

[1] X = supported; NA = not applicable

22.6 UART registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Tab 22.6-1 UART maps

offset	Register	Reset value	Description
USART1: USART1_BA = 0x4001_3800 USART2: USART2_BA = 0x4000_4400 USART3: USART3_BA = 0x4000_4800 UART4: UART4_BA = 0x4000_4C00 UART5: UART5_BA = 0x4000_5000			
0x00	USART_SR	0x0000_00C0	USART Status register
0x04	USART_DR	0x0000_0000	USART Data register
0x08	USART_BRR	0x0000_0000	USART Baud rate register
0x0C	USART_CR1	0x0000_0000	USART Control register 1
0x10	USART_CR2	0x0000_0000	USART Control register 2
0x14	USART_CR3	0x0000_0000	USART Control register 3
0x18	USART_GTPR	0x0000_0000	USART Guard time and prescaler register

22.6.1 USART Status register(USART_SR)

Register	Address offset	Access	Reset value	Description
USART_SR	0x00	RW	0x0000_0000	USART Status register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						CTS	LBD
7	6	5	4	3	2	1	0
TXE	TC	RXNE	IDLE	ORE	NE	FE	PE

USART Status register(UART_SR)bit description

Bit	Access	Description
[31:10]	R	Reserved, forced by hardware to 0.
[9]	RC_W0	**CTS:** CTS flag This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software (by writing it to 0). An interrupt is generated if CTSIE=1 in the USART_CR3 register. 0: No change occurred on the CTS status line 1: A change occurred on the CTS status line **Note:** *This bit is not available for UART4 & UART5*
[8]	RC_W0	**LBD:** LIN break detection flag This bit is set by hardware when the LIN break is detected. It is cleared by software (by writing it to 0). An interrupt is generated if LBDIE = 1 in the USART_CR2 register. 0: LIN Break not detected 1: LIN break detected **Note:** *An interrupt is generated when LBD=1 if LBDIE=1*
[7]	R	**TXE:** Transmit data register empty This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register. It is cleared by a write to the USART_DR register. 0: Data is not transferred to the shift register 1: Data is transferred to the shift register **Note:** *This bit is used during single buffer transmission*
[6]	RC_W0	**TC:** Transmission complete This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE=1 in the USART_CR1 register. It is cleared by a software sequence (a read from the USART_SR register followed by a write to the USART_DR register). The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for multibuffer communication. 0: Transmission is not complete 1: Transmission is complete

[5]	RC_W0	**RXNE:** Read data register not empty This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_DR register. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. This clearing sequence is recommended only for multibuffer communication. 0: Data is not received 1: Received data is ready to be read
[4]	R	**IDLE:** IDLE line detected This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the IDLEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register). 0: No Idle Line is detected 1: Idle Line is detected **Note:** *The IDLE bit will not be set again until the RXNE bit has been set itself (i.e. a new idle line occurs).*
[3]	R	**ORE:** Overrun error This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RXNE=1. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register). 0: No Overrun error 1: Overrun error is detected **Note:** *When this bit is set, the RDR register content will not be lost but the shift register will be overwritten. An interrupt is generated on ORE flag in case of Multi Buffer communication if the EIE bit is set.*
[2]	R	**NE:** Noise error flag This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register). 0: No noise is detected 1: Noise is detected **Note:** *This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupting interrupt is generated on NE flag in case of Multi Buffer communication if the EIE bit is set.*
[1]	R	**FE:** Framing error This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register). 0: No Framing error is detected 1: Framing error or break character is detected **Note:** *This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. If the word currently being transferred causes both frame error and overrun error, it will be transferred and only the ORE bit will be set. An interrupt is generated on FE flag in case of Multi Buffer communication if the EIE bit is set*

[0]	R	**PE:** Parity error This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by a software sequence (a read to the status register followed by a read to the USART_DR data register). The software must wait for the RXNE flag to be set before clearing the PE bit. An interrupt is generated if PEIE = 1 in the USART_CR1 register. 0: No parity error 1: Parity error

22.6.2 USART Data register(USART_DR)

Register	Address offset	Access	Reset value	Description
USART_DR	0x04	RW	0x0000_0000	USART Data register

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

USART Data register(UART_DR)bit description

Bit	Access	Description
[31:9]	R	Reserved, forced by hardware to 0.
[8:0]	RW	**DR[8:0]:** Data value Contains the Received or Transmitted data character, depending on whether it is read from or written to. The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR) The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 1). The RDR register provides the parallel interface between the input shift register and the internal bus. When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity. When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

22.6.3 USART Baud rate register(USART_BRR)

Note: *The baud counters stop counting if the TE or RE bits are disabled respectively.*

Register	Address offset	Access	Reset value	Description
USART_BRR	0x08	RW	0x0000_0000	USART Baud rate register

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

USART Baud rate register(UART_BRR)bit description

Bit	Access	Description
[31:16]	R	Reserved, forced by hardware to 0
[15:4]	RW	**DIV_Mantissa[11:0]:** mantissa of USARTDIV These 12 bits define the mantissa of the USART Divider (USARTDIV)
[3:0]	RW	**DIV_Fraction[3:0]:** fraction of USARTDIV These 4 bits define the fraction of the USART Divider (USARTDIV)

22.6.4 USART Control register 1(USART_CR1)

Register	Address offset	Access	Reset value	Description
USART_CR1	0x0C	RW	0x0000_0000	USART Control register 1

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		UE	M	WAKE	PCE	PS	PEIE
7	6	5	4	3	2	1	0
TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK

USART Control register 1(UART_CR1)bit description

Bit	Access	Description
[31:14]	R	Reserved, forced by hardware to 0.
[13]	RW	**UE:** USART enable When this bit is cleared the USART prescalers and outputs are stopped and the end of the current byte transfer in order to reduce power consumption. This bit is set and cleared by software. 0: USART prescaler and outputs disabled 1: USART enabled
[12]	RW	**M:** Word length This bit determines the word length. It is set or cleared by software. 0: 1 Start bit, 8 Data bits, n Stop bit 1: 1 Start bit, 9 Data bits, n Stop bit **Note:** *The M bit must not be modified during a data transfer (both transmission and reception)*
[11]	RW	**WAKE:** Wakeup method This bit determines the USART wakeup method, it is set or cleared by software. 0: Idle Line 1: Address Mark

[10]	RW	**PCE:** Parity control enable This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission). 0: Parity control disabled 1: Parity control enabled
[9]	RW	**PS:** Parity selection This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte. 0: Even parity 1: Odd parity
[8]	RW	**PEIE:** PE interrupt enable This bit is set and cleared by software. 0: Interrupt is inhibited 1: A USART interrupt is generated whenever PE=1 in the USART_SR register
[7]	RW	**TXEIE:** TXE interrupt enable This bit is set and cleared by software. 0: Interrupt is inhibited 1: A USART interrupt is generated whenever TXE=1 in the USART_SR register
[6]	RW	**TCIE:** Transmission complete interrupt enable This bit is set and cleared by software. 0: Interrupt is inhibited 1: A USART interrupt is generated whenever TC=1 in the USART_SR register
[5]	RW	**RXNEIE:** RXNE interrupt enable This bit is set and cleared by software. 0: Interrupt is inhibited 1: A USART interrupt is generated whenever ORE=1 or RXNE=1 in the USART_SR register
[4]	RW	**IDLEIE:** IDLE interrupt enable This bit is set and cleared by software. 0: Interrupt is inhibited 1: A USART interrupt is generated whenever IDLE=1 in the USART_SR register
[3]	RW	**TE:** Transmitter enable This bit enables the transmitter. It is set and cleared by software. 0: Transmitter is disabled 1: Transmitter is enabled **Note:** 1. *During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word, except in Smartcard mode.* 2. *When TE is set there is a 1 bit-time delay before the transmission starts.*
[2]	RW	**RE:** Receiver enable This bit enables the receiver. It is set and cleared by software. 0: Receiver is disabled 1: Receiver is enabled and begins searching for a start bit

[1]	RW	**RWU:** Receiver wakeup This bit determines if the USART is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wakeup sequence is recognized. 0: Receiver in active mode 1: Receiver in mute mode **Note:** 1. *Before selecting Mute mode (by setting the RWU bit) the USART must first receive a data byte, otherwise it cannot function in Mute mode with wakeup by Idle line detection.* 2. *In Address Mark Detection wakeup configuration (WAKE bit=1) the RWU bit cannot be modified by software while the RXNE bit is set.*
[0]	RW	**SBK:** Send break This bit set is used to send break characters. It can be set and cleared by software. It should be set by software, and will be reset by hardware during the stop bit of break. 0: No break character is transmitted 1: Break character will be transmitted

22.6.5 USART Control register 2(USART_CR2)

Register	Address offset	Access	Reset value	Description
USART_CR2	0x10	RW	0x0000_0000	USART Control register 2

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL
7	6	5	4	3	2	1	0
Reserved	LBDIE	LBDL	Reserved	ADD[3:0]			

USART Control register 2(UART_CR2)bit description

Bit	Access	Description
[31:15]	R	Reserved, forced by hardware to 0.
[14]	RW	**LINEN:** LIN mode enable This bit is set and cleared by software. 0: LIN mode disabled 1: LIN mode enabled The LIN mode enables the capability to send LIN Synch Breaks (13 low bits) using the SBK bit in the USART_CR1 register, and to detect LIN Sync breaks.
[13:12]	RW	**STOP[1:0]:** STOP bits These bits are used for programming the stop bits. 00: 1 Stop bit 01: 0.5 Stop bit 10: 2 Stop bits 11: 1.5 Stop bit **Note:** *The 0.5 Stop bit and 1.5 Stop bit are not available for UART4 & UART5.*

[11]	RW	**CLKEN:** Clock enable This bit allows the user to enable the CK pin. 0: CK pin disabled 1: CK pin enabled **Note:** *This bit is not available for UART4 & UART5*
[10]	RW	**CPOL:** Clock polarity This bit allows the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship 0: Steady low value on CK pin outside transmission window. 1: Steady high value on CK pin outside transmission window. **Note:** *This bit is not available for UART4 & UART5.*
[9]	RW	**CPHA:** Clock phase This bit allows the user to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship 0: The first clock transition is the first data capture edge. 1: The second clock transition is the first data capture edge. **Note:** *This bit is not available for UART4 & UART5.*
[8]	RW	**LBCL:** Last bit clock pulse This bit allows the user to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode. 0: The clock pulse of the last data bit is not output to the CK pin 1: The clock pulse of the last data bit is output to the CK pin **Note:** 1. *The last bit is the 8th or 9th data bit transmitted depending on the 8 or 9 bit format selected by the M bit in the USART_CR1 register.* 2. *This bit is not available for UART4 & UART5.*
[7]	R	Reserved, forced by hardware to 0.
[6]	RW	**LBDIE:** LIN break detection interrupt enable Break interrupt mask (break detection using break delimiter). 0: Interrupt is inhibited 1: An interrupt is generated whenever LBD=1 in the USART_SR register
[5]	RW	**LBDL:** lin break detection length This bit is for selection between 11 bit or 10 bit break detection. 0: 10 bit break detection 1: 11 bit break detection
[4]	R	Reserved, forced by hardware to 0.
[3:0]	RW	**ADD[3:0]:** Address of the USART node This bit-field gives the address of the USART node. This is used in multiprocessor communication during mute mode, for wake up with address mark detection.

22.6.6 USART Control register 3(USART_CR3)

Register	Address offset	Access	Reset value	Description
USART_CR3	0x14	RW	0x0000_0000	USART Control register 3

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					CTSIE	CTSE	RTSE
7	6	5	4	3	2	1	0
DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE

USART Control register 3(UART_CR3)bit description

Bit	Access	Description
[31:11]	R	Reserved, forced by hardware to 0.
[10]	RW	**CTSIE:** CTS interrupt enable 0: Interrupt is inhibited 1: An interrupt is generated whenever CTS=1 in the USART_SR register **Note:** *This bit is not available for UART4 & UART5.*
[9]	RW	**CTSE:** CTS enable 0: CTS hardware flow control disabled 1: CTS mode enabled, data is only transmitted when the CTS input is asserted (tied to 0). If the CTS input is deasserted while a data is being transmitted, then the transmission is completed before stopping. If a data is written into the data register while CTS is deasserted, the transmission is postponed until CTS is asserted. **Note:** *This bit is not available for UART4 & UART5.*
[8]	RW	**RTSE:** RTS enable 0: RTS hardware flow control disabled 1: RTS interrupt enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is asserted (tied to 0) when a data can be received. **Note:** *This bit is not available for UART4 & UART5.*
[7]	RW	**DMAT:** DMA enable transmitter This bit is set/reset by software 1: DMA mode is enabled for transmission 0: DMA mode is disabled for transmission **Note:** *This bit is not available for UART5*
[6]	RW	**DMAR:** DMA enable receiver This bit is set/reset by software 1: DMA mode is enabled for reception 0: DMA mode is disabled for reception **Note:** *This bit is not available for UART5*
[5]	RW	**SCEN:** Smartcard mode enable This bit is used for enabling Smartcard mode. 0: Smartcard Mode disabled 1: Smartcard Mode enabled **Note:** *This bit is not available for UART4 & UART5.*
[4]	RW	**NACK:** Smartcard NACK enable 0: NACK transmission in case of parity error is disabled 1: NACK transmission during parity error is enabled **Note:** *This bit is not available for UART4 & UART5.*

[3]	RW	**HDSEL:** Half-duplex selection Selection of Single-wire Half-duplex mode 0: Half duplex mode is not selected 1: Half duplex mode is selected
[2]	RW	**IRLP:** IrDA low-power This bit is used for selecting between normal and low-power IrDA modes 0: Normal mode 1: Low-power mode
[1]	RW	**IREN:** IrDA mode enable This bit is set and cleared by software. 0: IrDA disabled 1: IrDA enabled
[0]	RW	**EIE:** Error interrupt enable Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise error (FE=1 or ORE=1 or NE=1 in the USART_SR register) in case of Multi Buffer Communication (DMAR=1 in the USART_CR3 register) 0: Interrupt is inhibited 1: An interrupt is generated whenever DMAR=1 in the USART_CR3 register and FE=1 or ORE=1 or NE=1 in the USART_SR register.

22.6.7 USART Guard time and prescaler register (USART_GTPR)

Register	Address offset	Access	Reset value	Description
USART_GTPR	0x18	RW	0x0000_0000	USART Guard time and prescaler register

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

USART Guard time and prescaler register(UART_GTPR)bit description

Bit	Access	Description
[31:16]	R	Reserved, forced by hardware to 0.
[15:8]	RW	**GT[7:0]:** Guard time value This bit-field gives the Guard time value in terms of number of baud clocks. This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value. **Note:** *This bit is not available for UART4 & UART5.*

[7:0]	RW	**PSC[7:0]:** Prescaler value - In IrDA Low-power mode: PSC[7:0] = IrDA Low-Power Baud Rate Used for programming the prescaler for dividing the system clock to achieve the low-power frequency: The source clock is divided by the value given in the register (8 significant bits): 00000000: Reserved - do not program this value 00000001: divides the source clock by 1 00000010: divides the source clock by 2 …… –In normal IrDA mode: PSC must be set to 00000001. –In Smartcard mode: PSC[4:0]: Prescaler value Used for programming the prescaler for dividing the system clock to provide the smartcard clock. The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency: 00000: Reserved - do not program this value 00001: divides the source clock by 2 00010: divides the source clock by 4 00011: divides the source clock by 6 …… **Note:** 1. *Bits [7:5] have no effect if Smartcard mode is used.* 2. *This bit is not available for UART4 & UART5.*

23 Integrated Circuit Interface(I2C)

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. I2C bus uses two serial lines: a serial data line, SDA, and a serial clock line, SCL.

The I2C interface implements standard I2C protocol with standard-mode, fast-mode and high-speed mode as well as SMBus (system management bus) and PMBus (power management bus). It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

The device includes two I2C blocks: I2C1 and I2C2.

23.1 I2C Characteristics

- Two-wire I2C serial interface –consists of a serial data line (SDA) and a serial clock line (SCL)
- Both master and slave functions with the same interface.
- Three speeds:
  - Standard mode (0 to 100 Kb/s)
  - Fast mode ($\leqslant$ 400 Kb/s) or fast mode plus ($\leqslant$ 1000 Kb/s)
- 7- or 10-bit addressing
- 7- or 10-bit combined format transfers
- Bulk transmit mode
- Ignores CBUS addresses (an older ancestor of I2C that used to share the I2C bus)
- Transmit and receive buffers
- Handles Bit and Byte waiting at all bus speeds
- Interrupt or polled-mode operation
- Component parameters for configurable software driver support
- Programmable SDA hold time
- Support DMA mode
- Bus clear feature
- Device ID feature
- SMBus/PMBus Support
- SMBus Slave detects and responds to ARP commands.
- Support SMBUS Alarm and timeout detection

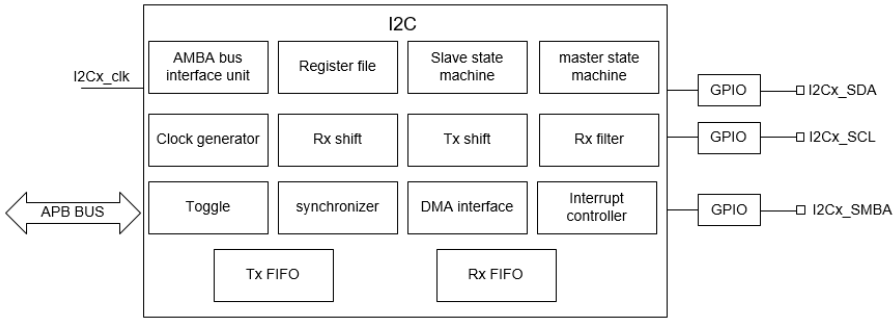
23.2 I2C implementation

Tab 23.2-1 I2C1 and I2C2 Features

I2C features	I2C1	I2C2
7 bit addressing	◯	◯
10 bit addressing	◯	◯
Standard mode	◯	◯
Fast mode/Fast Plus Mode	◯	◯
High speed mode	X	◯
SMBUS	◯	X

23.3 Block Diagram

Fig 23.3-1 Block Diagram



23.4 I2C Function Overview

23.4.1 Mode Selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, the device operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, it supports multi-master operation.

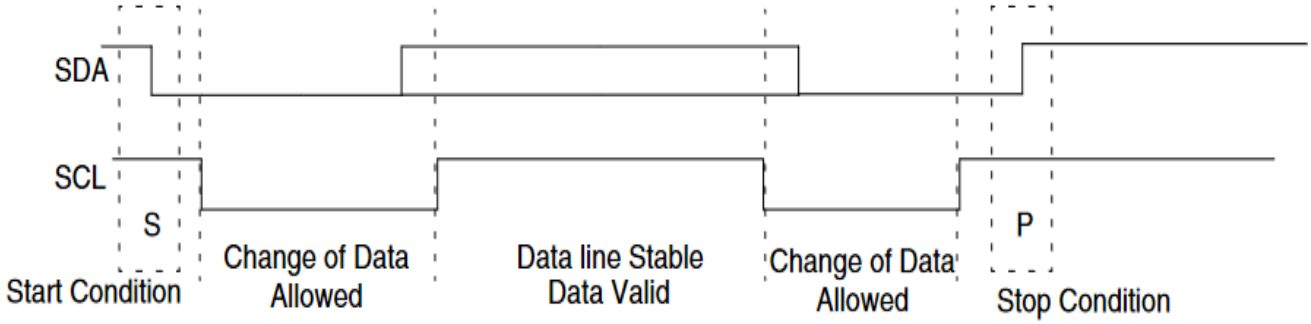
Start condition

When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1.

Stop condition

When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1. The figure below shows the start/stop condition in detail.

Fig 23.4-1 START and STOP Condition



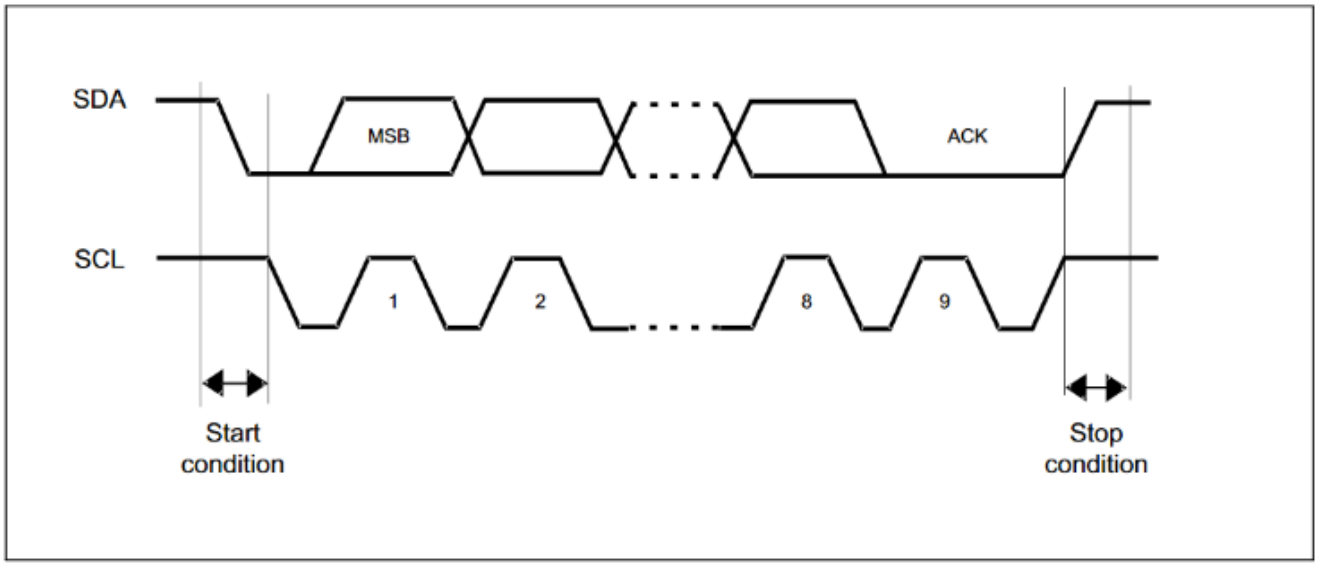
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The Reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure 23.4-2

Fig 23.4-2 START and STOP Condition



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by

software.

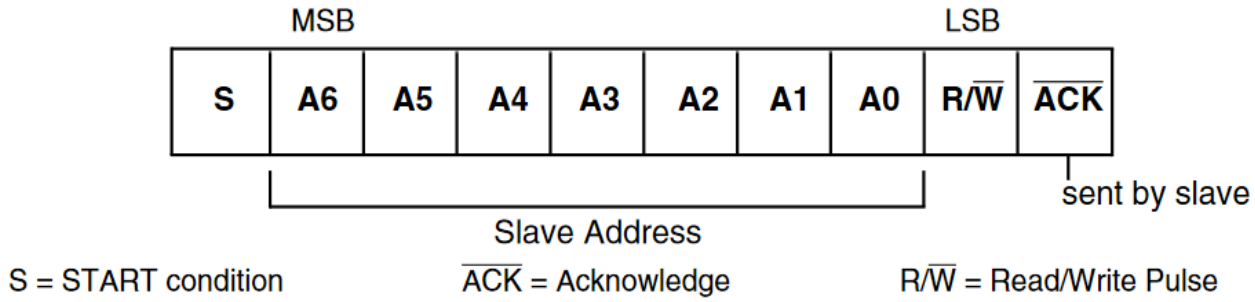
23.5 Addressing Slave Protocol

There are two address formats: the 7-bit address format and the 10-bit address format.

7-bit Address Format

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit (bit 0) is the R/W bit as shown in Figure 23.5-1. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave.

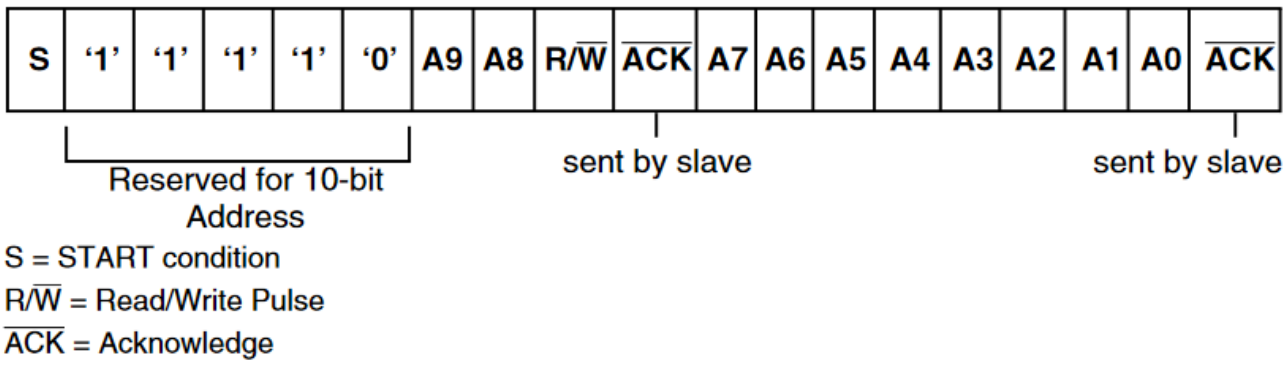
Fig 23.5-1 START and STOP Condition



10-bit Address Format

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves address bits 9:8, and the LSB bit (bit0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. Figure 23.5-2 shows the 10-bit address format.

Fig 23.5-2 START and STOP Condition



The Table below defines the special purpose and Reserved first byte addresses

Tab 23.5-1 I2C/SMBus Definition of Bits in First Bye

Slave Address	R/W Bit	Description
0000 000	0	General Call Address. I2C places the data in the receive buffer and issues a General Call interrupt.
0000 000	1	START byte.
0000 001	X	CBUS address. I2C ignores these accesses.
0000 010	X	Reserved.
0000 011	X	Reserved
0000 1XX	X	High speed master mode.
1111 1XX	X	Reserved
1111 0XX	X	10-bit slave addressing
0001 000	X	SMBus Host
0001 100	X	SMBus Alert Response Address
1100 001	X	SMBus Device Default Address

I2C does not restrict you from using these Reserved addresses. However, if you use these Reserved addresses, you may run into incompatibilities with other I2C components.

23.5.1 Transmitting and Receiving Protocol

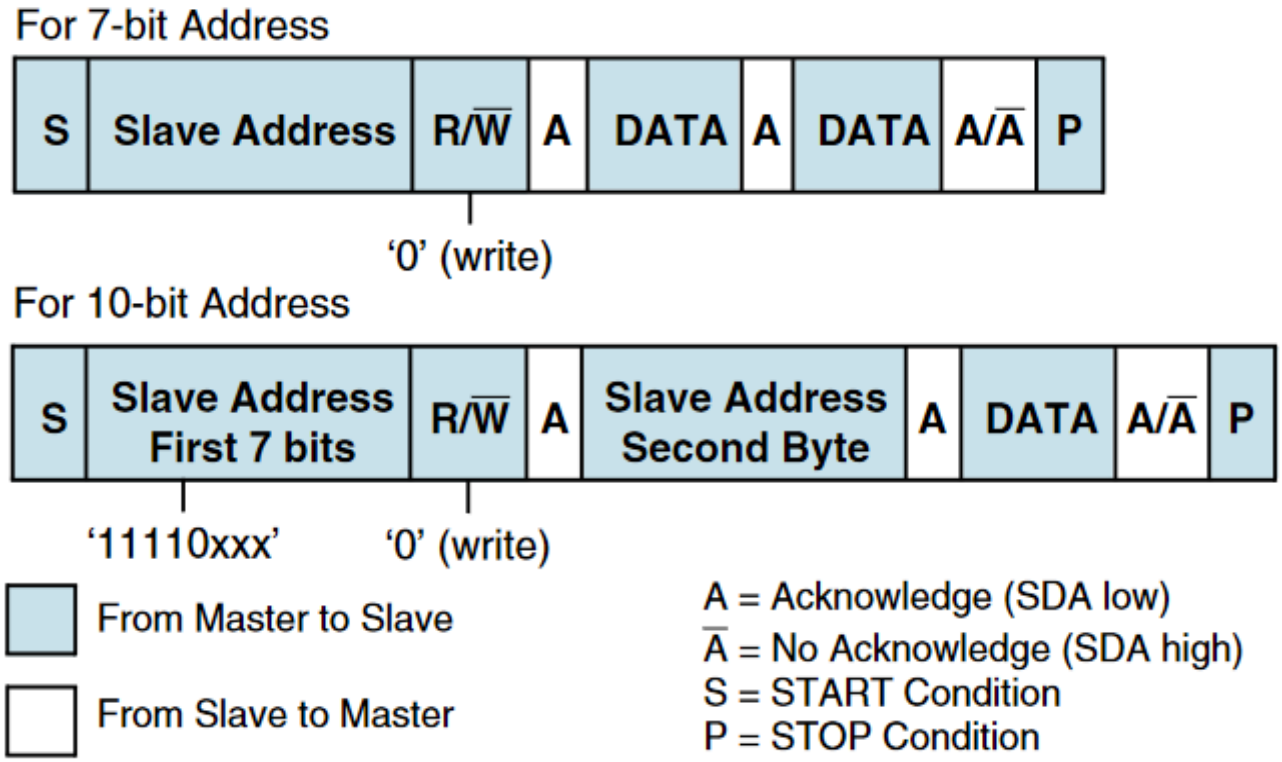
The master can initiate data transmission and reception to/from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests from the master to either transmit data or receive data to/from the bus, acting as either a slave-transmitter or slave-receiver, respectively.

Master-Transmitter and Slave-Receiver

All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer.

If the master-transmitter is transmitting data as shown in Figure 23.5-3, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

Fig 23.5-3 Master-Transmitter Protocol

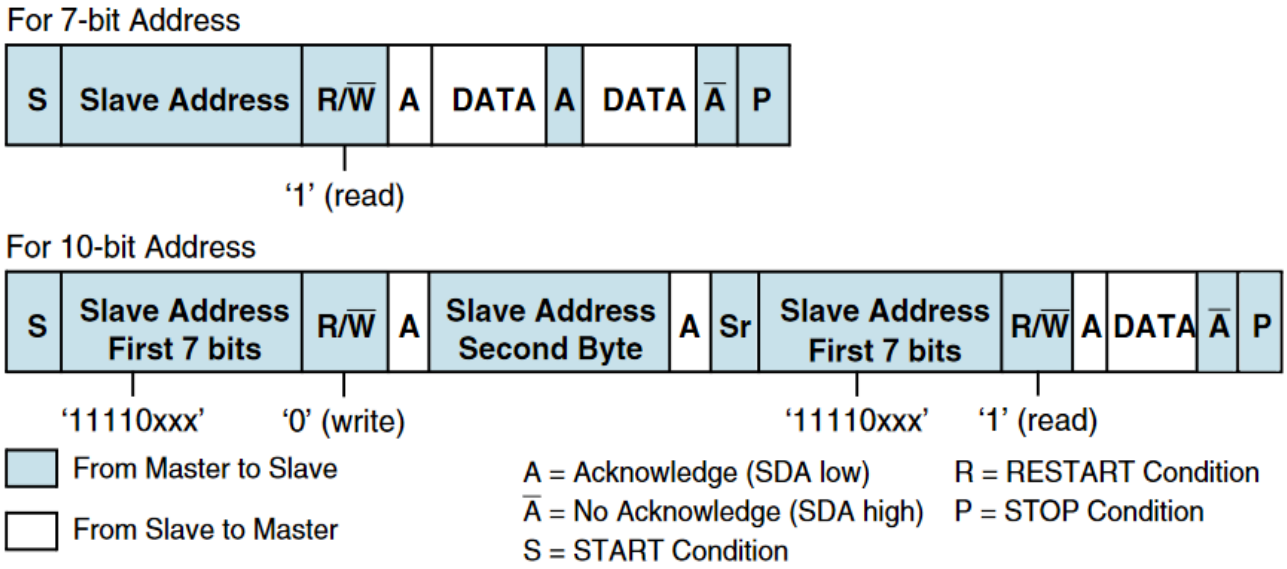


Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in Figure 23.5-4, then the master responds to the slave-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte.

The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the master can issue a STOP condition. When a master does not want to relinquish the bus with a STOP condition, the master can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse.

Fig 23.5-4 Master-Receiver Protocol

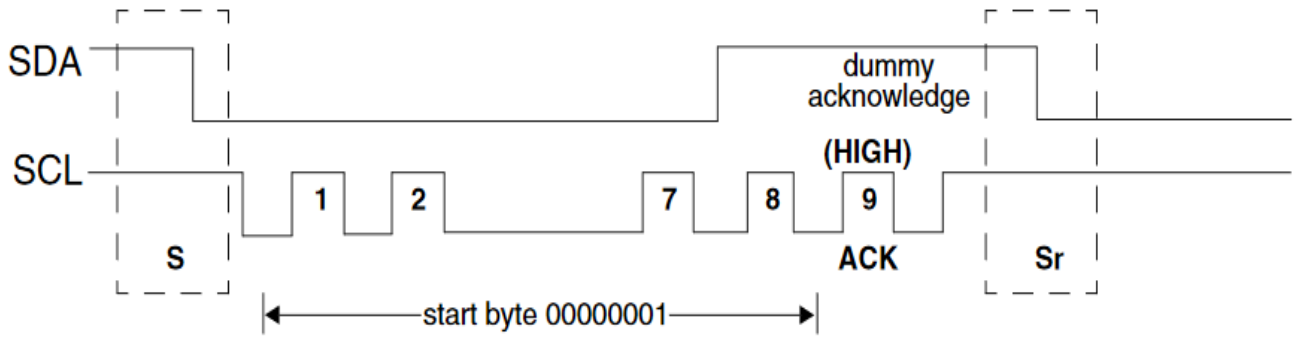


23.6 START BYTE Transfer Protocol

The START BYTE transfer protocol is set up for systems that do not have an on-board dedicated I2C hardware module. When the I2C is addressed as a slave, it always samples the I2C bus at the highest speed supported so that it never requires a START BYTE transfer. However, when I2C is a master, it supports the generation of START BYTE transfers at the beginning of every transfer in case a slave device requires it.

This protocol consists of seven zeros being transmitted followed by a 1, as illustrated in Figure 23.6-1. This allows the processor that is polling the bus to under-sample the address phase until 0 is detected. Once the microcontroller detects a 0, it switches from the under sampling rate to the correct rate of the master.

Fig 23.6-1 START BYTE Transfer



The START BYTE procedure is as follows:、

- Master generates a START condition.

- Master transmits the START byte (0000 0001).
- Master transmits the ACK clock pulse. (Present only to conform with the byte handling format used on the bus)
- No slave sets the ACK signal to 0.
- Master generates a RESTART (R) condition. A hardware receiver does not respond to the START BYTE because it is a Reserved address and reset after the RESTART condition is generated.

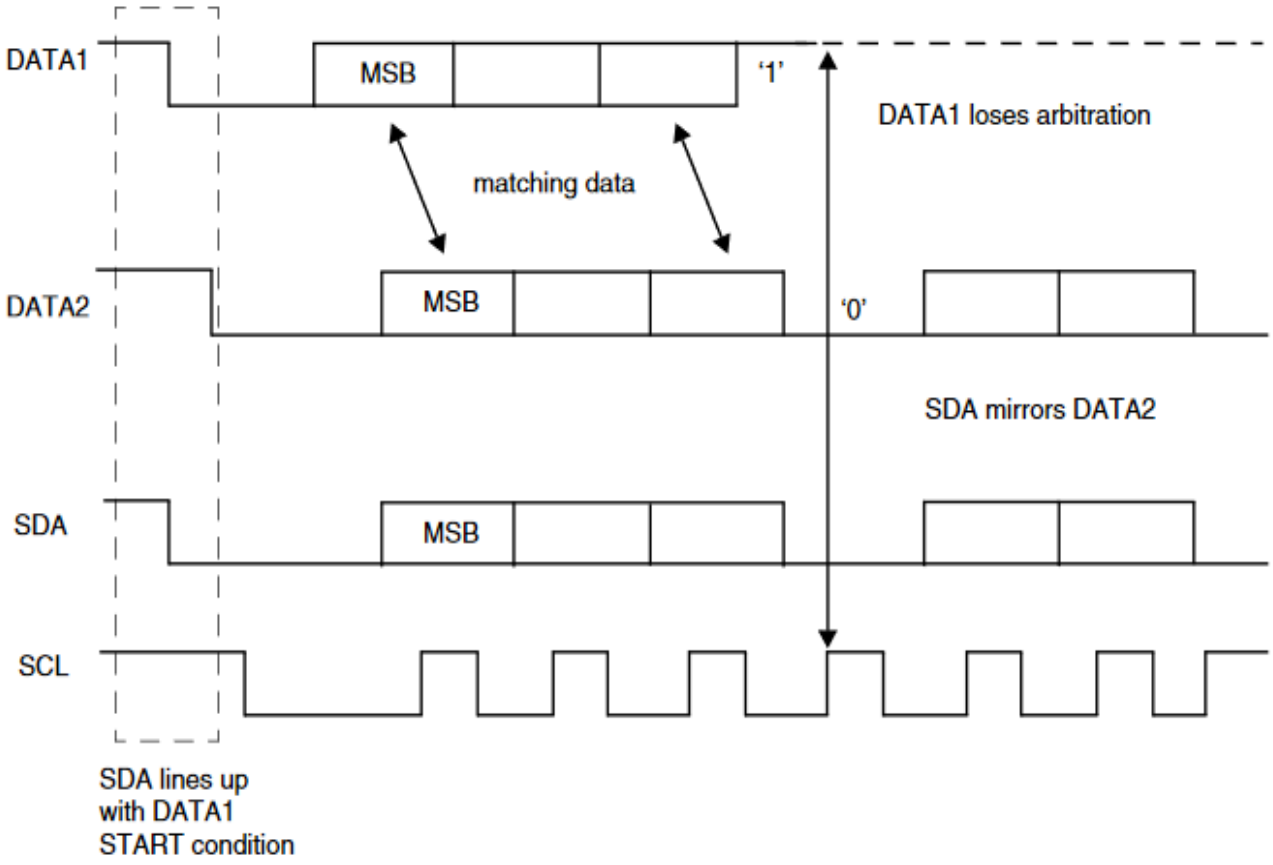
23.6.1 Multiple Master Arbitration

The I2C bus protocol allows multiple masters to reside on the same bus. If there are two masters on the same I^2C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating a START condition at the same time. Once a master (for example, a microcontroller) has control of the bus, no other master can take control until the first master sends a STOP condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is 1. The master, which transmits a 1 while the other master transmits 0, loses arbitration and turns off its data output stage. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase.

Upon detecting that it has lost arbitration to another master, the I2C will stop generating SCL (ic_clk_oe). Figure 26-9 illustrates the timing of when two masters are arbitrating on the bus.

Fig 23.6-2 Multiple Master Arbitration



For high-speed mode, the arbitration cannot go into the data phase because each master is programmed with a unique high-speed master code. This 8-bitcode is defined by the system designer and is set by writing to the High Speed Master Mode Code Address Register, IC_HS_MADDR. Because the codes are unique, only one master can win arbitration, which occurs by the end of the transmission of the high-speed master code.

Control of the bus is determined by address or master code and data sent by competing masters, so there is no central master nor any order of priority on the bus.

Arbitration is not allowed between the following conditions:

- A RESTART condition and a data bit
- A STOP condition and a data bit
- A RESTART condition and a STOP condition

Slaves are not involved in the arbitration process.

23.6.2 Slave Mode Operation

This section discusses slave mode procedures in detail.

Initial Configuration

To use the I2C as a slave mode, perform the following steps:

- Disable the I2C macro by writing a ‘0’ to bit 0 of the IC_ENABLE register.
- Write to the IC_SAR register (bits 9:0) to set the slave address. This is the address to which the I2C macro responds.
- Write to the IC_CON register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Enable the I2C macro in slave-only mode by writing a‘0’into bit 6 (IC_SLAVE_DISABLE) and a ‘0’ to bit 0 (MASTER_MODE).
- Enable the I2C macro by writing a ‘1’ in bit 0 of the IC_ENABLE register.

23.6.3 Slave-Transmitter Operation for a Single Byte

When another I2C master device on the bus addresses this I2C macro and requests data, the I2C macro acts as a slave-transmitter and the following steps occur:

- The other I2C master device initiates an I2C transfer with an address that matches the slave address in the IC_SAR register of this I2C macro.
- The I2C macro acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.
- The I2C macro asserts the RD_REQ interrupt (bit 5 of the IC_RAW_INTR_STAT register) and holds the SCL line low. It is in a wait state until software responds. If the RD_REQ interrupt has been masked, due to IC_INTR_MASK[5] register (M_RD_REQ bit field) being set to 0, then it is recommended that a hardware and/or software timing routine be used to instruct the CPU to perform periodic reads of the IC_RAW_INTR_STAT register.
  - Reads that indicate IC_RAW_INTR_STAT[5] (R_RD_REQ bit field) being set to 1 must be treated as the equivalent of the RD_REQ interrupt being asserted.
  - Software must then act to satisfy the I2C transfer.
  - The timing interval used should be in the order of 10 times the fastest SCL clock period the I2C macro can handle. For example, for 400 kb/s, the timing interval is 25us.
- If there is any data remaining in the Tx FIFO before receiving the read request, then the I2C macro asserts a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register) to flush the old data from the TX FIFO. If the TX_ABRT interrupt has been masked, due to of IC_INTR_MASK[6] register (M_TX_ABRT bit field) being set to 0, then it is recommended that re-using the timing routine (described in the previous step), or a similar one, be used to read the IC_RAW_INTR_STAT register.
  - Reads that indicate bit 6 (R_TX_ABRT) being set to 1 must be treated as the equivalent of the TX_ABRT interrupt being asserted.
  - There is no further action required from software.
  - The timing interval used should be similar to that described in the previous step for the IC_RAW_INTR_STAT[5] register.
- Software writes to the IC_DATA_CMD register with the data to be written (by writing a ‘0’ in bit 8).
- Software must clear the RD_REQ and TX_ABRT interrupts (bits 5 and 6, respectively) of the IC_RAW_INTR_STAT register before proceeding.

- If the RD_REQ and/or TX_ABRT interrupts have been masked, then clearing of the IC_RAW_INTR_STAT register will have already been performed when either the R_RD_REQ or R_TX_ABRT bit has been read as 1.
- The I2C macro releases the SCL and transmits the byte.
- The master may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

Note: *Whenever a TX_ABRT event occurs, it is necessary for software to read the IC_CLR_TX_ABRT register before attempting to write into the Tx FIFO. See register IC_RAW_INTR_STAT for more details.*

Slave-Receiver Operation for a Single Byte

When another I2C master device on the bus addresses this I2C macro and is sending data, the I2C macro acts as a slave-receiver and the following steps occur:

- The other I2C master device initiates an I2C transfer with an address that matches its slave address in the IC_SAR register.
- The I2C macro acknowledges the sent address and recognizes the direction of the transfer to indicate that the I2C macro is acting as a slave-receiver.
- The I2C macro receives the transmitted byte and places it in the receive buffer.
- The I2C macro asserts the RX_FULL interrupt (IC_RAW_INTR_STAT[2] register). If the RX_FULL interrupt has been masked, due to setting IC_INTR_MASK[2] register to 0 or setting IC_TX_TL to a value larger than 0, then it is recommended that a timing routine (described in "Slave-Transmitter Operation for a Single Byte") be implemented for periodic reads of the IC_STATUS register. Reads of the IC_STATUS register, with bit 3 (RFNE) set at 1, must then be treated by software as the equivalent of the RX_FULL interrupt being asserted.
- Software may read the byte from the IC_DATA_CMD register (bits 7:0).
- The other master device may hold the I2C bus by issuing a RESTART condition, or release the bus by issuing a STOP condition.

Slave-Transfer Operation for Bulk Transfers

In the standard I2C protocol, all transactions are single byte transactions and the programmer responds to a remote master read request by writing one byte into the slave's TXFIFO. When a slave (slave-transmitter) is issued with a read request (RD_REQ) from the remote master (master-receiver), at a minimum there should be at least one entry placed into the slave-transmitter's TXFIFO. This I2C macro is designed to handle more data in the TXFIFO so that subsequent read requests can take that data without raising an interrupt to get more data. Ultimately, this eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the TXFIFO.

This mode only occurs when the I2C macro is acting as a slave-transmitter. If the remote master acknowledges the data sent by the slave-transmitter and there is no data in the slave's TXFIFO, the I2C macro holds the SCL line low while it raises the read request interrupt (RD_REQ) and waits for data to be written into the TXFIFO before it can be sent to the remote master.

If the RD_REQ interrupt is masked, due to bit 5 (M_RD_REQ) of the IC_INTR_STAT register being set to 0, then it is recommended that a timing routine be used to activate periodic reads of the IC_RAW_INTR_STAT register. Reads of IC_RAW_INTR_STAT that return bit 5 (R_RD_REQ) set to 1 must be treated as the equivalent of the RD_REQ interrupt referred to in this section. This timing routine is similar to that described in “Slave-Transmitter Operation for a Single Byte”.

The RD_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the TxFIFO. During the transmission of these bytes to the master, if the master acknowledges the last byte. Then the slave must raise the RD_REQ again because the master is requesting for more data.

If the programmer knows in advance that the remote master is requesting a packet of n bytes, then when another master addresses this I2C macro and requests data, the TxFIFO could be written with n number bytes and the remote master receives it as a continuous stream of data. For example, the I2C macro slave continues to send data to the remote master as long as the remote master is acknowledging the data sent and there is data available in the TxFIFO. There is no need to hold the SCL line low or to issue RD_REQ again. If the remote master is to receive n bytes from the I2C macro but the programmer wrote a number of bytes larger than n to the TxFIFO, then when the slave finishes sending the requested n bytes, it clears the TxFIFO and ignores any excess bytes.

The I2C macro generates a transmit abort (TX_ABRT) event to indicate the clearing of the Tx FIFO in this example. At the time an ACK/NACK is expected, if a NACK is received, then the remote master has all the data it wants. At this time, a flag is raised within the slave’s state machine to clear the leftover data in the Tx FIFO. This flag is transferred to the processor bus clock domain where the FIFO exists and the contents of the TxFIFO is cleared at that time.

23.6.4 Master Mode Operation

This section discusses master mode procedures in detail.

Initial Configuration

To use the I2C macro as a master, perform the following steps:

- Disable the macro by writing 0 to bit 0 of the IC_ENABLE register.
- Write to the IC_CON register to set the maximum speed mode supported (bits 2:1) and the desired speed of the I2C macro master-initiated transfers, either 7-bit or 10-bit addressing (bit 4). Ensure that bit 6 (IC_SLAVE_DISABLE) is written with a ‘1’ and bit 0 (MASTER_MODE) is written with a ‘1’.
- Write to the IC_TAR register the address of the I2C device to be addressed (bits 9:0). This register also indicates whether a General Call or a START BYTE command is going to be performed by I2C.
- Only applicable for high-speed mode transfers. Write to the IC_HS_MADDR register the desired master code for the I2C macro. The master code is programmer-defined.
- Enable the DW_apb_i2c by writing a 1 to bit 0 of the IC_ENABLE register.

- Now write transfer direction and data to be sent to the IC_DATA_CMD register. If the IC_DATA_CMD register is written before the DW_apb_i2c is enabled, the data and commands are lost as the buffers are kept cleared when DW_apb_i2c is disabled.

This step generates the START condition and the address byte on the I2C macro. Once the I2C macro is enabled and there is data in the TX FIFO, the I2C macro starts reading the data.

Master Transmit and Master Receive

The I2C macro supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I2C Rx/Tx Data Buffer and Command Register (IC_DATA_CMD). The CMD bit [8] should be written to 0 for write operations. Subsequently, a read command may be issued by writing "don't cares" to the lower byte of the IC_DATA_CMD register, and a 1 should be written to the CMD bit. The I2C macro (as a master) continues to initiate transfers as long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty—depending on the value of IC_EMPTYFIFO_HOLD_MASTER_EN, the master either inserts a STOP condition after completing the current transfers, or it checks to see if IC_DATA_CMD[9] is set to 1.

- If set to 1, it issues a STOP condition after completing the current transfer.
- If set to 0, it holds SCL low until next command is written to the transmit FIFO.

23.6.5 Disable I2C

To disable I2C, user can clear bit 0 of IC_ENABLE. Then the user should poll the register IC_ENABLE_STATUS to unambiguously determine when the hardware has completely shut down.

23.6.6 Aborting I2C Transfers

The ABORT control bit of the IC_ENABLE register allows the software to relinquish the I2C bus before completing the issued transfer commands from the Tx FIFO. In response to an ABORT request, the controller issues the STOP condition over the I2C bus, followed by Tx FIFO flush. Aborting the transfer is allowed only in master mode of operation. To abort the I2C transfers, follow the steps below:

- Stop filling the Tx FIFO (IC_DATA_CMD) with new commands.
- When operating in DMA mode, disable the transmit DMA by setting TDMAE to 0.
- Set bit 1 of the IC_ENABLE register (ABORT) to 1.
- Wait for the M_TX_ABRT interrupt.
- Read the IC_TX_ABRT_SOURCE register to identify the source as ABRT_USER_ABRT.

23.6.7 Spike Suppression

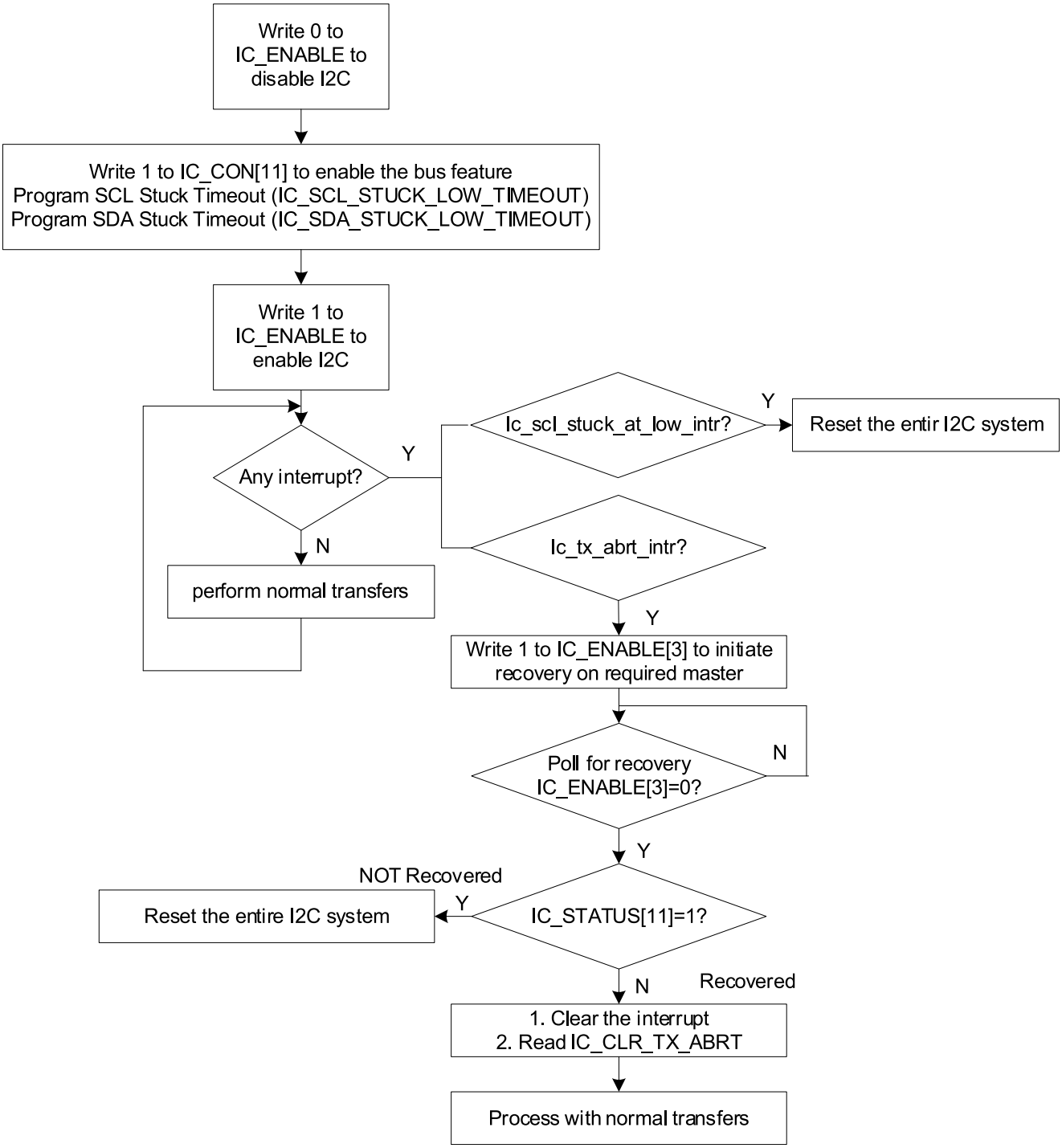
The I2C macro contains programmable spike suppression logic, and the logic is based on counters that monitor the input signals (SCL and SDA), checking if they remain stable for a predetermined amount of ic_clk cycles before they are sampled internally. There is one separate counter for each signal (SCL and SDA). The number of ic_clk cycles can be programmed by the user and should be calculated taking into account the frequency of ic_clk and the relevant spike length specification.

Based on I2C bus protocol, the maximum spike length for SS/FS mode is 50ns, and is 10ns for HS.

23.6.8 I2C Bus Clear

The I2C macro supports the bus clear feature that provides graceful recovery of data (SDA) and clock (SCL) lines during unlikely events in which either the clock or data line is stuck at LOW

Fig 23.6-3 Bus Clear Flow



23.6.9 Device ID

A Device ID field is an optional 3-byte read-only (24 bits) word, which provides the following information:

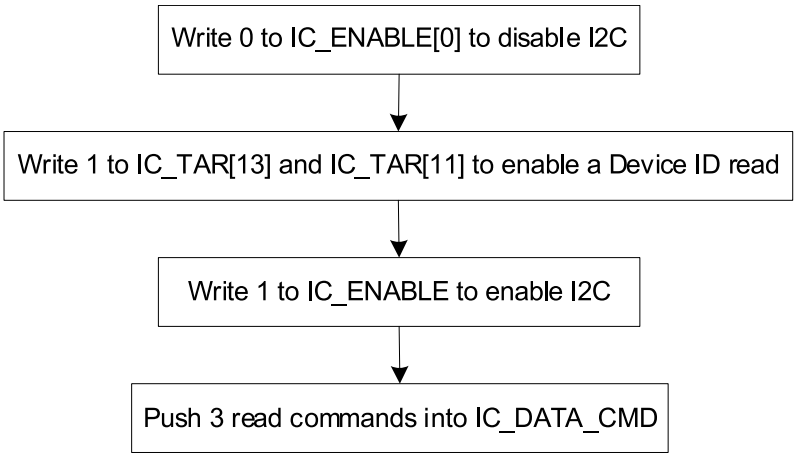
- Twelve bits with the manufacturer’ s name, which is unique for every manufacturer.
- Nine bits with the part identification, which is assigned by the manufacturer.
- Three bits with the die revision, which is assigned by the manufacturer.

For reading the Device ID of a particular slave, the master can follow the procedure in figure 26-11. The Device ID that is read will be available in RX FIFO, which can be read using IC_DATA_CMD register.

In case of a slave, the user can read the Device ID of the slave using IC_DEVICE_ID register.

Device ID is not supported for 10-bit addressing and High Speed transfers (HS mode)

Fig 23.6-4 Reading Device ID



23.6.10 SMBUS Protocol

A typical SMBus device will have a set of commands by which data can be read and written. All commands are one byte long while their arguments and return values can vary in length. In accordance with the SMBus specification, the most significant bit (MSB) is transferred first. There are eleven possible command protocols for any given device. These commands are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write, and Block Write-Block Read Process Call.

SMBus protocols for message transactions are generally different from I2C data transfer commands. It is still possible to program a SMBus master to deliver I2C data transfer commands. The following table describes the derivation of SMBus Bus Protocols through Tx-FIFO commands in this macro.

In the SMBus Master mode, all the receive data bytes will be available in Rx-FIFO. In the SMBus Slave mode, all the bus protocol command codes and data bytes will received in the Rx-FIF0 and read request data bytes must be sent using Tx-FIFO, similar to the I2C mode.

The macro slave can be enabled to receive only Quick command through enabling the SLAVE_QUICK_CMD_EN bit in the IC_CON Register. Whenever this bit is selected the slave only receives quick commands and

will not accept other Bus Protocols. The macro slave issues the SMBUS_QUICK_DET interrupt upon receiving the QUICK command.

SMBus introduces a Packet Error checking Mechanism through appending PEC Byte at the end of the Bus Protocol. This can be achieved through adding an extra command (PEC byte) while transferring and decoding it while receiving by the software.

23.6.11 SMBUS Address Resolution Protocol

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device by the Host. This feature allows the devices to be 'hot-plugged' in to the system.

SMBus introduces a 128-bit Unique Device ID (UDID) for each device in the system to isolate each device for the purpose of address assignment. This macro uses the IC_SMBUS_UDID_MSB parameter for upper constant 96 bits and 'IC_SMBUS_ARP_UDID_LSB' register for lower variable 32 bits of the UDID.

This macro uses the PERSISTANT_SLV_ADDR_EN register bit in IC_CON register to indicate whether it supports persistent slave address.

This master can issue general and directed Address Resolution Protocol (ARP) commands to assign the dynamic address for the slaves in the SMBus system.

Note:

The macro slave hardware handles the generation, detection, and NACKing of the wrong PEC (CRC8 C(X)=X8+X2+X1+1) for the ARP Commands, but don't handle the PEC for Non-ARP commands.

The macro master hardware does not handle PEC for both APR and non-ARP commands

23.6.12 SMBus Alert

This macro includes the SMBus alter signal (SMBALERT#) specified by the SMBus standard. It can be used by simple devices to request the attention of the host.

In slave mode, user can set SMBUS_ALERT_CTRL bit (IC_ENABLE[18]) to enable it. Once asserted by user, the user waits for alert response address to be sent by master. Upon receiving it, contents of IC_SAR[7:0] register are sent to the master. When successful, this macro clears the SMBUS_ALERT_CTRL bit to stop sending the alert.

If working as host, user needs to service the alert detect interrupt by sending read byte command with Alert Response Address. Current alarm status can be read from SMBUS_ALERT_STATUS bit (IC_STATUS[20]).

23.6.13 SDA Hold Time

It is defined in I2C protocol that SDA needs to meet a certain holding time. The user can configure the IC_ SDA_HOLD register to dynamically adjust the hold time of SDA. When the transmission speed of I2C bus changes, it is necessary to adjust the SDA dynamic holding time at the same time.

IC_SDA_TX_HOLD uses IC_ SDA_Hold [15:0] to adjust the holding time of SDA in transmission mode.

However, in the transmission mode, there is a minimum SDA holding time. As the main transmitter, the minimum SDA holding time is 1 ic_clk. As a slave transmitter, the minimum SDA holding time is SPKLEN + 7 ic_clk period (SPKLEN refers to IC_FS_SPKLEN or IC_ HS_ SPKLEN)。

23.6.14 IC_CLK Frequency Configuration

When this macro is configured as a Standard (SS), Fast (FS)/Fast-Mode Plus (FM+), the *CNT registers must be set before any I2C bus transaction can take place in order to ensure proper I/O timing. The *CNT registers are:

- IC_SS_SCL_HCNT
- IC_SS_SCL_LCNT
- IC_FS_SCL_HCNT
- IC_FS_SCL_LCNT

When configure this registers, there are some requirements:

- IC_SS_SCL_LCNT and IC_FS_SCL_LCNT register values must be larger than IC_FS_SPKLEN + 7.
- IC_SS_SCL_HCNT and IC_FS_SCL_HCNT register values must be larger than IC_FS_SPKLEN + 5.

Tab 23.6-1

mode	ic_clk freq(MHz)	smallest value of IC_*SPK_LEN	SCL low time	SCL low configure	SCL high time	SCL high configure
SS	2.7	1	4.7 us	12	5.2 us	6
FS	12	1	1.33 us	15	1.16 us	6
HS(400pf)	51	1	333ns	16	274 ns	6
HS(100pf)	105.4	1	161ns	16	132 ns	6

23.6.15 DMA interfaces

This macro supports DMA transmission and can send DMA application to DMAC. The user can write the IC_DMA_CR to turn on the send DMA and receive DMA functions. When the data in the transmit buffer is less than or equal to the value defined in IC_ DMA_TDLR, a transmit DMA requests will be generated. When the number of data in RXFIFO is larger than or equal to the number specified in the IC_DMA_RDLR, a receive DMA request is generated.

Before DMA transmission, users need to configure DMAC.

23.6.16 Interrupts

For the interrupt types in I2C mode and SMBus mode, please refer to the corresponding register (I2C_RAW_INTR_STAT and SMBUS_RAW_INTR_STAT) for detail info.

23.7 I2C Registers

Tab 23.7-1 I2C registers map

Offset	Register	Reset value	Description
I2C1_BA = 0x4000_5400 I2C2_BA = 0x4000_5800			
0x00	I2C_CON	0x0000_0024	I2C Control Register
0x04	I2C_TAR	0x0000_0055	I2C Target Address Register
0x08	I2C_SAR	0xXXXX_X055	I2C Slave Address Register
0x10	I2C_DATA_CMD	0x0000_0000	I2C Data and Command Register
0x14	I2C_SS_SCL_HCNT	0x0000_0028	Standard Speed I2C Clock SCL High Count Register
0x18	I2C_SS_SCL_LCNT	0x0000_002F	Standard Speed I2C Clock SCL Low Count Register
0x1C	I2C_FS_SCL_HCNT	0x0000_0006	Full Speed I2C Clock SCL High Count Register.
0x20	I2C_FS_SCL_LCNT	0x0000_000D	Full Speed I2C Clock SCL Low Count Register
0x2C	I2C_INTR_STAT	0x0000_0000	I2C Interrupt Status Register
0x30	I2C_INTR_MASK	0x0000_48FF	I2C Interrupt Mask Register
0x34	I2C_RAW_INTR_STAT	0x0000_0000	I2C RAW Interrupt Status Register
0x38	I2C_RX_TL	0x0000_0000	I2C Receive FIFO Threshold Register
0x3C	I2C_TX_TL	0x0000_0000	I2C Transmit FIFO Threshold Register
0x40	I2C_CLR_INTR	0x0000_0000	I2C Clear Combined and Individual Interrupt Register
0x44	I2C_CLR_RX_UNDER	0x0000_0000	Clear RX_UNDER Interrupt Register
0x48	I2C_CLR_RX_OVER	0x0000_0000	Clear RX_OVER Interrupt Register
0x4C	I2C_CLR_TX_OVER	0x0000_0000	Clear TX_OVER Interrupt Register
0x50	I2C_CLR_RD_REQ	0x0000_0000	Clear RD_REQ Interrupt Register
0x54	I2C_CLR_TX_ABRT	0x0000_0000	Clear TX_ABRT Interrupt Register
0x58	I2C_CLR_RX_DONE	0x0000_0000	Clear RX_DONE Interrupt Register
0x5C	I2C_CLR_ACTIVITY	0x0000_0000	Clear ACTIVITY Interrupt Register
0x60	I2C_CLR_STOP_DET	0x0000_0000	Clear STOP_DET Interrupt Register
0x64	I2C_CLR_START_DET	0x0000_0000	Clear START_DET Interrupt Register
0x68	I2C_CLR_GEN_CALL	0x0000_0000	Clear GEN_CALL Interrupt Register
0x6C	I2C_ENABLE	0x0000_0000	I2C Enable Register
0x70	I2C_STATUS	0x0010_0006	I2C_STATUS Register
0x74	I2C_TXFLR	0x0000_0000	I2C Transmit FIFO Level Register
0x78	I2C_RXFLR	0x0000_0000	I2C Receive FIFO Level Register
0x7C	I2C_SDA_HOLD	0x0000_0001	I2C SDA Hold Time Length Register
0x80	I2C_TX_ABRT_SOURCE	0x0000_0000	I2C Transmit Abort Source Register
0x88	I2C_DMA_CR	0x0000_0000	DMA Control Register
0x8C	I2C_DMA_TDLR	0x0000_0000	DMA Transmit Data Level Register
0x90	I2C_DMA_RDLR	0x0000_0000	DMA Receive Data Level Register
0x94	I2C_SDA_SETUP	0x0000_0064	I2C SDA Setup Register
0x98	I2C_ACK_GENERAL_CALL	0x0000_0001	I2C ACK General Call Register
0x9C	I2C_ENABLE_STATUS	0x0000_0000	I2C Enable Status Register
0xA0	I2C_FS_SPKLEN	0x0000_0001	I2C SS, FS spike suppression limit
0xAC	I2C_SCL_STUCK_AT_LOW_TIMEOUT	0xFFFF_FFFF	I2C SCL Stuck at Low Timeout register

0xB0	I2C_SDA_STUCK_AT_LOW_TIMEOUT	0xFFFF_FFFF	I2C SDA Stuck at Low Timeout register
0xB4	I2C_CLR_SCL_STUCK_DET	0x0000_0000	Clear SCL Stuck at Low Detect interrupt Register
0xBC	SMBUS_CLK_LOW_SEXT	0xFFFF_FFFF	SMBus Slave Clock Extend Timeout register
0xC0	SMBUS_CLK_LOW_MEXT	0xFFFF_FFFF	SMBus Master Clock Extend Timeout register
0xC8	SMBUS_INTR_STAT	0x0000_0000	SMBus Interrupt Status Register
0xCC	SMBUS_INTR_MASK	0x0000_07FF	SMBus Interrupt Mask Register
0xD0	SMBUS_RAW_INTR_STAT	0x0000_0000	SMBus Raw Interrupt Status Register
0xD4	CLR_SMBUS_INTR	0x0000_0000	Clear SMBus Interrupt Register
0xD8	I2C_OPTIONAL_SAR	0x0000_0000	I2C Optional Slave Address Register
0xDC	SMBUS_UDID_LSB	0xFFFF_FFFF	SMBUS ARP UDID LSB Register

23.7.1 I2C Control Register.(I2C_CON)

Note: *It is only writable when IC_ENABLE[0]=0.*

Register	Address offset	Access	I2C1 Reset value	I2C2 Reset value	Description
I2C_CON	0x00	RW	0x0000_0024	0x0000_003E	I2C Control Register

31	30	29	28	27	26	25	24	23	22	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SMBUS_ARP_EN	Reserved					IC_SAR_DUAL_EN	BUS_CLEAR_FEATURE_CTRL	Reserved	RX_FIFO_FULL_HLD_CTRL	TX_EMPTY_CTRL	STOP_DET_IFADDRESSED	IC_SLAVE_DISABLE	IC_RESTART_EN	IC_10BITADDR_MASTER	IC_10BITADDR_SLAVE	SPEED		MASTER_MODE

I2C Registers(I2C_CON)bit description

Bit	Access	Description
[31:19]	R	Reserved
[18]	RW	**SMBUS_ARP_EN:** This bit is applicable only in Slave mode. Values: 0x1 (ENABLED): SMBus ARP control is enabled. 0x0 (DISABLED): SMBus ARP control is disabled.
[17:13]	R	Reserved
[12]	RW	**IC_SAR_DUAL_EN:** 1:Enable dual address in bit 7 slave mode
[11]	RW	**BUS_CLEAR_FEATURE_CTRL:** In Master mode: 1'b1: Bus Clear Feature is enabled. 1'b0: Bus Clear Feature is Disabled. In Slave mode, this register bit is not applicable
[10]	RW	**STOP_DET_IF_MASTER_ACTIV:** In Master mode: 1'b1: issues the STOP_DET interrupt only when master is active. 1'b0: issues the STOP_DET irrespective of whether master is active or not. In Slave mode, this register bit is not applicable

[9]	RW	**RX_FIFO_FULL_HLD_CTRL:** Values: 0x1 (ENABLED): Hold bus when RX_FIFO is full 0x0 (DISABLED): Overflow when RX_FIFO is full
[8]	RW	**TX_EMP_CTRL:** Values: 0x1 (ENABLED): Controlled generation of TX_EMPTY interrupt 0x0 (DISABLED): Default behavior of TX_EMPTY interrupt
[7]	RW	**STOP_DET_IFADDRESSED:** In slave mode: 1'b1: issues the STOP_DET interrupt only when it is addressed. 0'b0: issues the STOP_DET irrespective of whether it's addressed or not.
[6]	RW	**IC_SLV_DISABLE:** This bit controls whether I2C has its slave disabled. Values: 0x1 (SLAVE_DISABLED): Slave mode is disabled 0x0 (SLAVE_ENABLED): Slave mode is enabled **Note:** *Software should ensure that if this bit is written with 0, then bit 0 should also be written with a 0*
[5]	RW	**IC_RESTART_EN:** Determines whether RESTART conditions may be sent when acting as a master. When RESTART is disabled, the master is prohibited from performing the following functions: Sending a START BYTE • Performing any high-speed mode operation • High-speed mode operation • Performing direction changes in combined format mode • Performing a read operation with a 10-bit address 0x1 (ENABLED): Master restart enabled 0x0 (DISABLED): Master restart disabled
[4]	RW	**IC_10BITADDR_MASTER:** This controls whether the DW_apb_i2c starts its transfers in 7- or 10-bit addressing mode when acting as a master. Values: 0x1 (ADDR_10BITS): Master 10Bit addressing mode 0x0 (ADDR_7BITS): Master 7Bit addressing mode
[3]	RW	**IC_10BITADDR_SLAVE:** When acting as a slave, this bit controls whether the I2C macro responds to 7- or 10-bit addresses Values: 0x1 (ADDR_10BITS): Slave 10Bit addressing 0x0 (ADDR_7BITS): Slave 7Bit addressing
[2]	RW	**SPEED[1:0]:** These bits control at which speed the I2C macro operates, and they are only be valid when operating in master mode. Values: 0x1 (STANDARD): Standard Speed mode of operation 0x2 (FAST): Fast or Fast Plus mode of operation 0x3 (HIGH): High Speed mode of operation Others: Reserved
[0]	RW	**MASTER_MODE:** This bit controls whether the I2C macro master is enabled. Values: 0x1 (ENABLED): Master mode is enabled 0x0 (DISABLED): Master mode is disabled

23.7.2 I2C Target Address Register(I2C_TAR)

Note:

1. *It is only writable when IC_ENABLE[0]=0.*
2. *It is not needed to configure this register if the macro works in slave mode.*

Register	Address offset	Access	Reset value	Description
I2C_TAR	0x04	RW	0x0000_0055	I2C Target Address 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
Reserved															SMBUS_QUICK_CMD	Reserved				SPECIAL	GC_OP_START	IC_TAR[9:0]									

I2C Target Address Register(I2C_TAR)bit description

Bit	Access	Description
[31:17]	R	Reserved
[16]	RW	**SMBUS_QUICK_CMD:** If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a Quick command is to be performed. Values: 0x1 (ENABLED): Enable programming of QUICK-CMD transmission 0x0 (DISABLED): Disable programming of QUICK-CMD transmission
[15:12]	R	Reserved
[11]	RW	**SPECIAL:** This bit indicates whether software performs a Device-ID or General Call or START BYTE command. Values: 0x1 (ENABLED): Enables programming of GENERAL_CALL or START_BYTE transmission 0x0 (DISABLED): Disables programming of GENERAL_CALL or START_BYTE transmission
[10]	RW	**GC_OP_START:** If bit 11 (SPECIAL) is set to 1 and bit 13(Device-ID) is set to 0, then this bit indicates whether a General Call or START byte command is to be performed. Values: 0x1 (START_BYTE): START byte transmission 0x0 (GENERAL_CALL): GENERAL_CALL byte
[9:0]	RW	**IC_TAR[9:0]:** This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits.

23.7.3 I2C Slave Address Register(I2C_SAR)

Note: *It is only writable when IC_ENABLE[0]=0.*

Register	Address offset	Access	Reset value	Description
I2C_SAR	0x08	RW	0xXXXX_X055	I2C Slave Address 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
Reserved									IC_SAR_DUAL													IC_SAR[9:0]									

I2C Slave Address Register(I2C_SAR)bit description

Bit	Access	Description
[31:23]	R	Reserved
[22:16]	RW	**IC_SAR_DUAL:** Dual address, only for 7-bit slave mode
[15:10]	RW	Reserved
[9:0]	RW	**IC_SAR[9:0]:** The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used

23.7.4 I2C Data and Command Register(I2C_DATA_CMD)

Note: *It is only writable when IC_ENABLE[0]=0*

Register	Address offset	Access	Reset value	Description
I2C_DATA_CMD	0x10	RW	0x0000_0000	I2C Data and Command 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
Reserved																				FIRST_DATA_BYTE	RESTART	STOP	CMD	DAT[7:0]								

I2C Data and Command Register(I2C_DATA_CMD)bit description

Bit	Access	Description
[31:12]	R	Reserved
[11]	R	**FIRST_DATA_BYTE:** Indicates the first data byte received after the address phase for receive transfer in Master receiver or Slave receiver mode. Values: 0x1 (ACTIVE): Non sequential data byte received 0x0 (INACTIVE): Sequential data byte received
[10]	W	**RESTART:** This bit controls whether a RESTART is issued before the byte is sent or received. Values: 0x1 (ENABLE): a RESTART is issued before the data is sent/received (according to the value of CMD), regardless of whether or not the transfer direction is changing from the previous command 0x0 (DISABLE): a RESTART is issued only if the transfer direction is changing from the previous command

[9]	W	**STOP:** This bit controls whether a STOP is issued after the byte is sent or received. Values: 0x1 (ENABLE): STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty,the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus. 0x0 (DISABLE): STOP is not issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master continues the current transfer by sending/receiving data bytes according to the value of the CMD bit. If the Tx FIFO is empty, the master holds the SCL line low and stalls the bus until a new command is available in the Tx FIFO.
[8]	W	**CMD:** This bit controls whether a read or a write is performed. It controls only the direction when it acts as a master. When programming this bit, you should remember the following: attempting to perform a read operation after a General Call command has been sent results in a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register), unless bit 11 (SPECIAL) in the IC_TAR register has been cleared. If a ”1” is written to this bit after receiving a RD_REQ interrupt, then a TX_ABRT interrupt occurs. Values: 0x1 (READ): Master Read Command 0x0 (WRITE): Master Write Command
[7:0]	RW	**DAT[7:0]:** This register contains the data to be transmitted or received on the I2C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the macro. However, when you read this register, these bits return the value of data received on the I2C interface

23.7.5 Standard Speed I2C Clock SCL High Count Register(I2C_SS_SCL_HCNT)

Note: *It is only writable when IC_ENABLE[0]=0.*

Register	Address offset	Access	Reset value	Description
I2C_SS_SCL_HCNT	0x14	RW	0x0000_0028	Standard Speed I2C Clock SCL High Count 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
Reserved																IC_SS_SCL_HCNT[15:0]															

Standard Speed I2C Clock SCL High Count Register(I2C_SS_SCL_HCNT)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**IC_SS_SCL_HCNT[15:0]:** This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed, based on ic_clk period. The valid value is 6 ~ 65525.

23.7.6 Standard Speed I2C Clock SCL Low Count Register(I2C_SS_SCL_LCNT)

Note: *It is only writable when IC_ENABLE[0]=0*

Register	Address offset	Access	Reset value	Description
I2C_SS_SCL_LCNT	0x18	RW	0x0000_002F	Standard Speed I2C Clock SCL Low Count 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
Reserved																IC_SS_SCL_LCNT[15:0]															

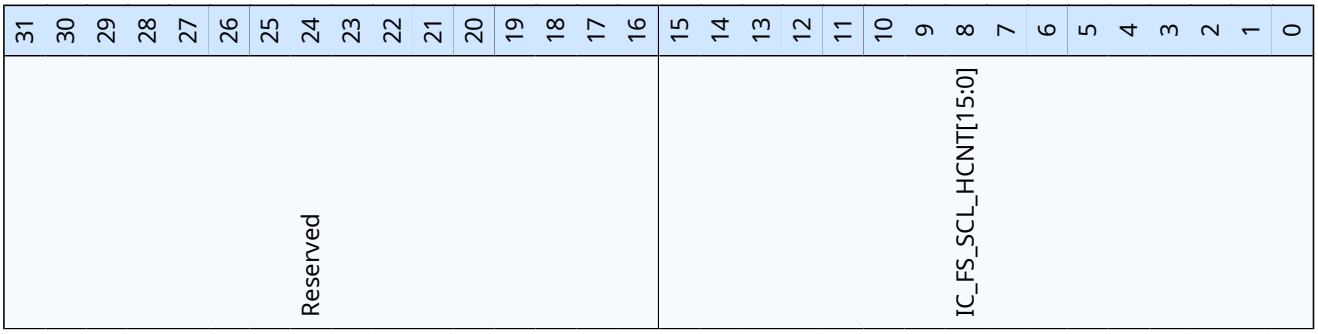
Standard Speed I2C Clock SCL Low Count Register(I2C_SS_SCL_LCNT)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**IC_SS_SCL_LCNT[15:0]:** This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low-period count for standard speed, based on ic_clk period. The minimum value is 8.

23.7.7 Full Speed I2C Clock SCL High Count Register(I2C_FS_SCL_HCNT)

Note: *It is only writable when IC_ENABLE[0]=0.*

Register	Address offset	Access	Reset value	Description
I2C_FS_SCL_HCNT	0x1C	RW	0x0000_0006	Full Speed I2C Clock SCL High Count Register



Full Speed I2C Clock SCL High Count Register(I2C_FS_SCL_HCNT)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**IC_FS_SCL_HCNT[15:0]:** This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed, based on ic_clk period. The minimum value is 6.

23.7.8 Full Speed I2C Clock SCL Low Count Register(I2C_FS_SCL_LCNT)

Note: *It is only writable when IC_ENABLE[0]=0*

Register	Address offset	Access	Reset value	Description
I2C_FS_SCL_LCNT	0x20	RW	0x0000_000D	Full Speed I2C Clock SCL Low Count Register

Full Speed I2C Clock SCL Low Count Register(I2C_SS_SCL_LCNT)bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**IC_FS_SCL_LCNT[15:0]:** This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed, based on ic_clk period. The minimum value is 8.

23.7.9 I2C Interrupt Status Register(I2C_INTR_STAT)

Note: *Each bit in this register has a corresponding mask bit in the IC_INTR_MASK register. These bits are cleared by reading the matching interrupt clear register. The unmasked raw versions of these bits are available in the IC_RAW_INTR_STAT register.*

Register	Address offset	Access	I2C1 Reset value	Description
I2C_INTR_STAT	0x2C	R	0x0000_0000	I2C Interrupt Status Register

31	30	29	28	27	26	25	24	23	22	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																R_SCL_STUCK_AT_LOW	Reserved		R_GEN_CALL	R_START_DET	R_STOP_DET	R_ACTIVITY	R_RX_DONE	R_TX_ABRT	R_RD_REQ	R_TX_EMPTY	R_TX_OVER	R_RX_FULL	R_RX_OVER	R_RX_UNDER

I2C Interrupt Status Register(I2C_INTR_STAT)bit description

Bit	Access	Description
[31:15]	R	Reserved
[14]	R	**R_SCL_STUCK_AT_LOW:** 0x1 (ACTIVE): R_SCL_STUCK_AT_LOW interrupt is active 0x0 (INACTIVE): R_SCL_STUCK_AT_LOW interrupt is inactive
[13:12]	R	Reserved
[11]	R	**R_GEN_CALL:** 0x1 (ACTIVE): R_GEN_CALL interrupt is active 0x0 (INACTIVE): R_GEN_CALL interrupt is inactive
[10]	R	**R_START_DET:** 0x1 (ACTIVE): R_START_DET interrupt is active 0x0 (INACTIVE): R_START_DET interrupt is inactive
[9]	R	**R_STOP_DET:** 0x1 (ACTIVE): R_STOP_DET interrupt is active 0x0 (INACTIVE): R_STOP_DET interrupt is inactive
[8]	R	**R_ACTIVITY:** 0x1 (ACTIVE): R_ACTIVITY interrupt is active 0x0 (INACTIVE): R_ACTIVITY interrupt is inactive
[7]	R	**R_RX_DONE:** 0x1 (ACTIVE): R_RX_DONE interrupt is active 0x0 (INACTIVE): R_RX_DONE interrupt is inactive
[6]	R	**R_TX_ABRT:** 0x1 (ACTIVE): R_TX_ABRT interrupt is active 0x0 (INACTIVE): R_TX_ABRT interrupt is inactive
[5]	R	**R_RD_REQ:** 0x1 (ACTIVE): R_RD_REQ interrupt is active 0x0 (INACTIVE): R_RD_REQ interrupt is inactive
[4]	R	**R_TX_EMPTY:** 0x1 (ACTIVE): R_TX_EMPTY interrupt is active 0x0 (INACTIVE): R_TX_EMPTY interrupt is inactive

[3]	R	**R_TX_OVER:** 0x1 (ACTIVE): R_TX_OVER interrupt is active 0x0 (INACTIVE): R_TX_OVER interrupt is inactive
[2]	R	**R_RX_FULL:** 0x1 (ACTIVE): R_RX_FULL interrupt is active 0x0 (INACTIVE): R_RX_FULL interrupt is inactive
[1]	R	**R_RX_OVER:** 0x1 (ACTIVE): R_RX_OVER interrupt is active 0x0 (INACTIVE): R_RX_OVER interrupt is inactive
[0]	R	**R_RX_UNSER:** 0x1 (ACTIVE): RX_UNDER interrupt is active 0x0 (INACTIVE): RX_UNDER interrupt is inactive

23.7.10 I2C Interrupt Mask Register(I2C_INTR_MASK)

Note: *These bits mask their corresponding interrupt status bits. This register is active low; a value of 0 masks the interrupt, whereas a value of 1 unmasks the interrupt.*

Register	Address offset	Access	I2C1 Reset value	Description
I2C_INTR_MASK	0x30	R	0x0000_48FF	I2C Interrupt Mask Register

31	30	29	28	27	26	25	24	23	22	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																M_SCL_STUCK_AT_LOW	Reserved		M_GEN_CALL	M_START_DET	M_STOP_DET	M_ACTIVITY	M_RX_DONE	M_TX_ABRT	M_RD_REQ	M_TX_EMPTY	M_TX_OVER	M_RX_FULL	M_RX_OVER	M_RX_UNDER

I2C Interrupt Mask Register(I2C_INTR_MASK)bit description

Bit	Access	Description
[31:15]	R	Reserved
[14]	RW	**M_SCL_STUCK_AT_LOW:** Mask bit for interrupt SCL_STUCK_AT_LOW
[13:12]	R	Reserved
[11]	RW	**M_GEN_CALL:** Mask bit for interrupt GEN_CALL
[10]	RW	**M_START_DET:** Mask bit for interrupt START_DET
[9]	RW	**M_STOP_DET:** Mask bit for interrupt STOP_DET
[8]	RW	**M_ACTIVITY:** Mask bit for interrupt ACTIVITY
[7]	RW	**M_RX_DONE:** Mask bit for interrupt RX_DONE
[6]	RW	**M_TX_ABRT:** Mask bit for interrupt TX_ABRT
[5]	RW	**M_RD_REQ:** Mask bit for interrupt RD_REQ
[4]	RW	**M_TX_EMPTY:** Mask bit for interrupt TX_EMPTY
[3]	RW	**M_TX_OVER:** Mask bit for interrupt TX_OVER
[2]	RW	**M_RX_FULL:** Mask bit for interrupt RX_FULL
[1]	RW	**M_RX_OVER:** Mask bit for interrupt RX_OVER
[0]	RW	**M_RX_UNDER:** Mask bit for interrupt RX_UNDER

23.7.11 I2C RAW Interrupt Status Register(I2C_RAW_INTR_STAT)

Note: *Unlike the IC_INTR_STAT register, these bits are not masked so they always show the true status. The interrupt bits are high active.*

Register	Address offset	Access	I2C1 Reset value	Description
I2C_RAW_INTR_STAT	0x34	R	0x0000_0000	I2C RAW Interrupt Status Register

31	30	29	28	27	26	25	24	23	22	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																SCL_STUCK_AT_LOW	Reserved		GEN_CALL	START_DET	STOP_DET	ACTIVITY	RX_DONE	TX_ABRT	RD_REQ	TX_EMPTY	TX_OVER	RX_FULL	RX_OVER	RX_UNDER

I2C RAW Interrupt Status Register(I2C_INTR_STAT)bit description

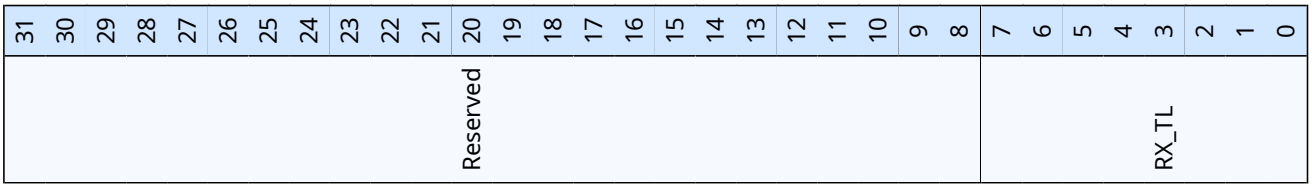
Bit	Access	Description
[31:15]	R	Reserved
[14]	R	**SCL_STUCK_AT_LOW:** Indicates whether the SCL Line is stuck at low for the IC_SCL_STUCK_LOW_TIMEOUT number of ic_clk periods.
[13:12]	R	Reserved
[11]	R	**GEN_CALL:** Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling the macro or when the CPU reads bit 0 of the IC_CLR_GEN_CALL register. The macro stores the received data in the Rx buffer
[10]	R	**START_DET:** Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether the macro is operating in slave or master mode
[9]	R	**STOP_DET:** Indicates whether a STOP condition has occurred on the I2C interface regardless of whether the macro is operating in slave or master mode. **In Slave Mode:** If IC_CON[7]=1'b1 (STOP_DET_IFADDRESSED), the STOP_DET interrupt will be issued only if slave is addressed If IC_CON[7]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET interrupt is issued irrespective of whether it is being addressed. **In Master Mode:** If IC_CON[10]=1'b1 (STOP_DET_IF_MASTER_ACTIVE), the STOP_DET interrupt will be issued only if Master is active. If IC_CON[10]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET interrupt will be issued irrespective of whether master is active or not.
[8]	R	**ACTIVITY:** This bit captures DW_apb_i2c activity and stays set until it is cleared. There are four ways to clear it: 1. Disabling the macro 2. Reading the IC_CLR_ACTIVITY register 3. Reading the IC_CLR_INTR register 4. System reset

[7]	R	**RX_DONE:** When the macro is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.
[6]	R	**TX_ABRT:** This bit indicates if the macro, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a 'transmit abort'. When this bit is set to 1, the IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places. **Note:** *The macro flushes/resets/empties only the TX_FIFO whenever there is a transmit abort caused by any of the events tracked by the IC_TX_ABRT_SOURCE register. The Tx FIFO remains in this flushed state until the register IC_CLR_TX_ABRT is read. Once this read is performed, the Tx FIFO is then ready to accept more data bytes from the APB interface.*
[5]	R	**RD_REQ:** This bit is set to 1 when the macro is acting as a slave and another I2C master is attempting to read data from it. The macro holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register
[4]	R	**TX_EMPTY:** The behavior of the TX_EMPTY interrupt status differs based on the TX_EMPTY_CTRL selection in the IC_CON register. When TX_EMPTY_CTRL = 0: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. When TX_EMPTY_CTRL = 1:This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register and the transmission of the address/data from the internal shift register for the most recently popped command is completed. It is automatically cleared by hardware when the buffer level goes above the threshold. When IC_ENABLE[0] is set to 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer any activity, then with ic_en=0, this bit is set to 0.
[3]	R	**TX_OVER:** Set during transmit if the transmit buffer is filled to IC_TX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.
[2]	R	**RX_FULL:** Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (IC_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the IC_ENABLE bit 0 is programmed with a 0, regardless of the activity that continues.
[1]	R	**RX_OVER:** Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an additional byte is received from an external I2C device. The DW_apb_i2c acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared. Note: If bit 9 of the IC_CON register (RX_FIFO_FULL_HLD_CTRL) is programmed to HIGH, then the RX_OVER interrupt never occurs, because the Rx FIFO never overflows.

[0]	R	**RX_UNDER:** Set if the processor attempts to read the receiver buffer when it is empty by reading from the IC_DATA_CMD register. If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.

23.7.12 I2C Receive FIFO Threshold Register(I2C_RX_TL)

Register	Address offset	Access	Reset value	Description
I2C_RX_TL	0x38	RW	0x0000_0000	I2C Receive FIFO Threshold Register

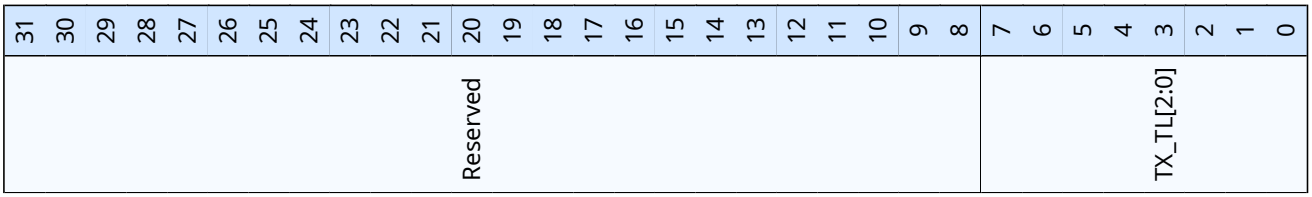


I2C Receive FIFO Threshold Register(I2C_RX_TL)bit description

Bit	Access	Description
[31:8]	R	Reserved
[7:0]	RW	**RX_TL[2:0]:** Receive FIFO Threshold Level. Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries

23.7.13 I2C Transmit FIFO Threshold Register(I2C_TX_TL)

Register	Address offset	Access	Reset value	Description
I2C_TX_TL	0x3C	RW	0x0000_0000	I2C Transmit FIFO Threshold Register

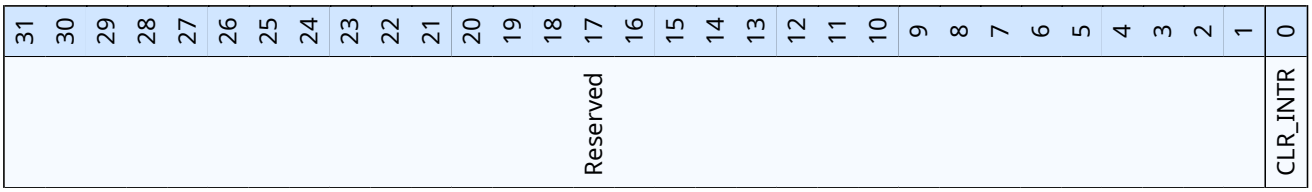


I2C Transmit FIFO Threshold Register(I2C_TX_TL)bit description

Bit	Access	Description
[31:8]	R	Reserved
[7:0]	RW	**TX_TL[2:0]:** Transmit FIFO Threshold Level. Controls the level of entries (or below) that trigger the TX_EMPTY interrupt (bit 4 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that it may not be set to value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 0 entries, and a value of 255 sets the threshold for 255 entries

23.7.14 I2C Clear Combined and Individual Interrupt Register(I2C_CLR_INTR)

Register	Address offset	Access	Reset value	Description
I2C_CLR_INTR	0x40	R	0x0000_0000	I2C Clear Combined and Individual Interrupt Register

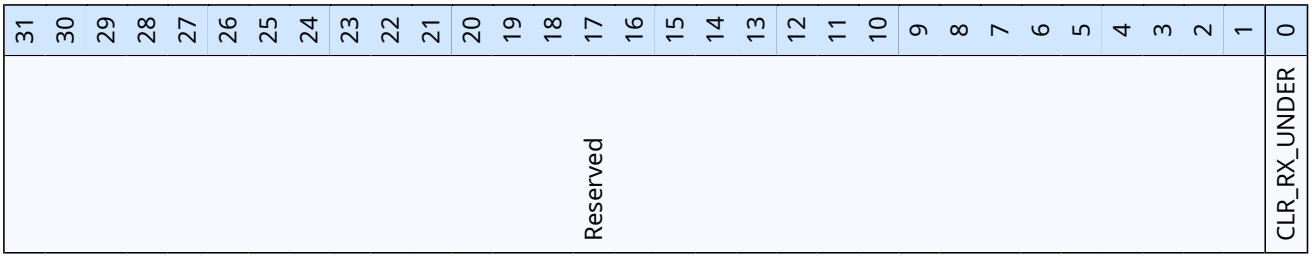


I2C Clear Combined and Individual Interrupt Register(I2C_CLR_INTR)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_INTR:** Read this register to clear the combined interrupt, all individual interrupts, and the IC_TX_ABRT_SOURCE register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE.

23.7.15 I2C Clear RX_UNDER Interrupt Register(I2C_CLR_RX_UNDER)

Register	Address offset	Access	Reset value	Description
I2C_CLR_RX_UNDER	0x44	R	0x0000_0000	I2C Clear RX_UNDER Interrupt Register

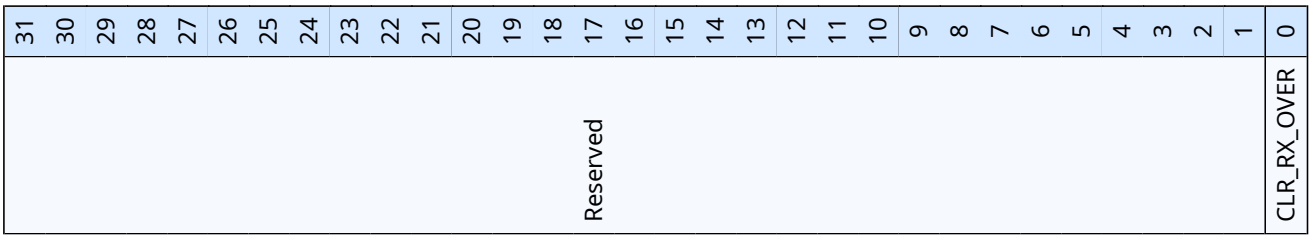


I2C Clear RX_UNDER Interrupt Register(I2C_CLR_RX_UNDER)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_RX_UNDER:** Read this register to clear the RX_UNDER interrupt (bit 0) of the IC_RAW_INTR_STAT register.

23.7.16 I2C Clear RX_OVER Interrupt Register(I2C_CLR_RX_OVER)

Register	Address offset	Access	Reset value	Description
I2C_CLR_RX_OVER	0x48	R	0x0000_0000	I2C Clear RX_OVER Interrupt Register

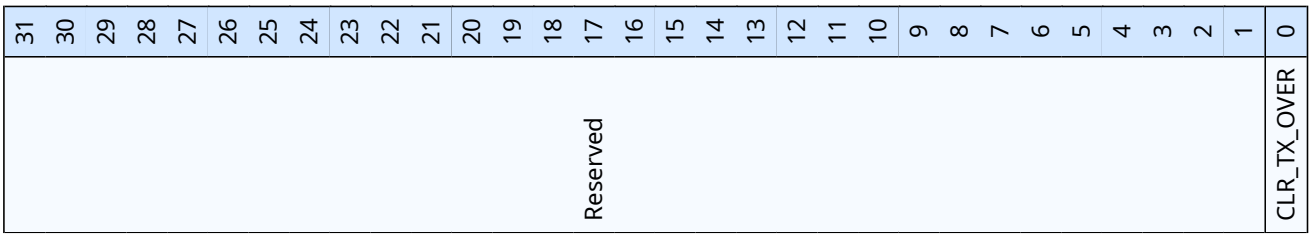


I2C Clear RX_OVER Interrupt Register(I2C_CLR_RX_OVER)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_RX_OVER:** Read this register to clear the RX_OVER interrupt (bit 1) of the IC_RAW_INTR_STAT register.

23.7.17 I2C Clear TX_OVER Interrupt Register(I2C_CLR_TX_OVER)

Register	Address offset	Access	Reset value	Description
I2C_CLR_TX_OVER	0x4C	R	0x0000_0000	I2C Clear TX_OVER Interrupt Register

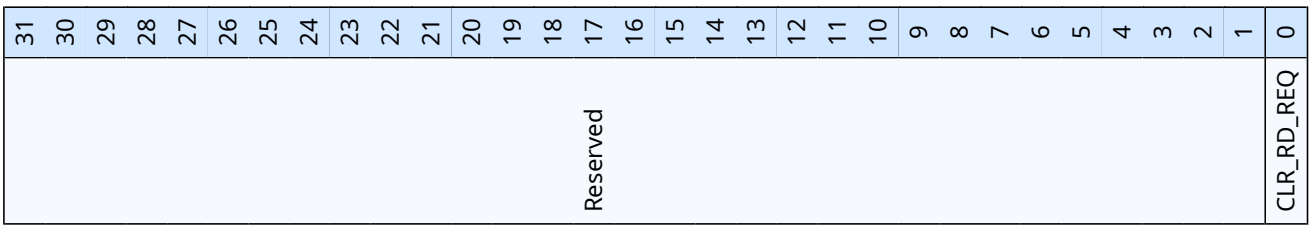


I2C Clear TX_OVER Interrupt Register(I2C_CLR_TX_OVER)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_TX_OVER:** Read this register to clear the TX_OVER interrupt (bit 3) of the IC_RAW_INTR_STAT register

23.7.18 I2C Clear RD_REQ Interrupt Register(I2C_CLR_RD_REQ)

Register	Address offset	Access	Reset value	Description
I2C_CLR_RD_REQ	0x50	R	0x0000_0000	I2C Clear RD_REQ Interrupt Register



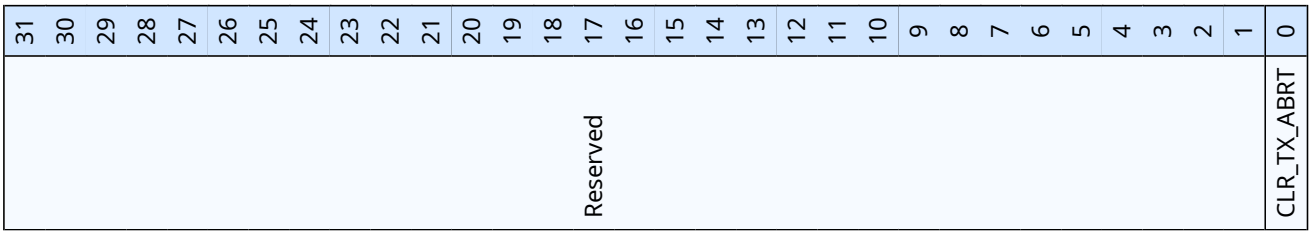
I2C Clear RD_REQ Interrupt Register(I2C_CLR_RD_REQ)bit description

Bit	Access	Description

[31:1]	R	Reserved
[0]	R	**CLR_RD_REQ:** Read this register to clear the RD_REQ interrupt (bit 5) of the IC_RAW_INTR_STAT register.

23.7.19 I2C Clear TX_ABRT Interrupt Registe(I2C_CLR_TX_ABRT)

Register	Address offset	Access	Reset value	Description
I2C_CLR_TX_ABRT	0x54	R	0x0000_0000	I2C Clear TX_ABRT Interrupt Registe

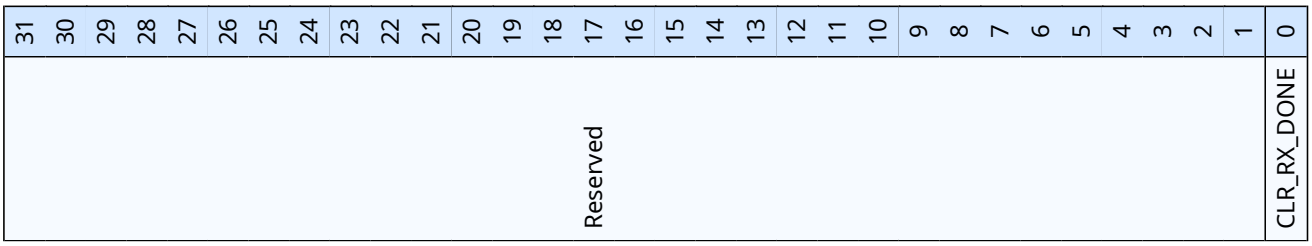


I2C Clear TX_ABRT Interrupt Registe(I2C_CLR_TX_ABRT)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_TX_ABRT:** Read this register to clear the TX_ABRT interrupt (bit 6) of the IC_RAW_INTR_STAT register, and the IC_TX_ABRT_SOURCE register. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE.

23.7.20 I2C Clear RX_DONE Interrupt Register(I2C_CLR_RX_DONE)

Register	Address offset	Access	Reset value	Description
I2C_CLR_RX_DONE	0x58	R	0x0000_0000	I2C Clear RX_DONE Interrupt Register

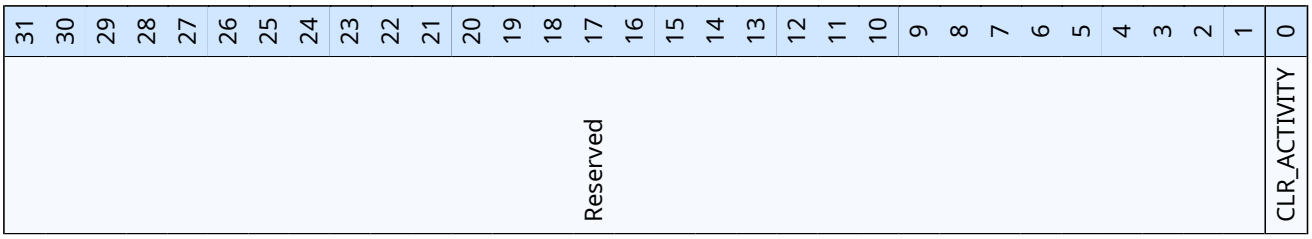


I2C Clear RX_DONE Interrupt Register(I2C_CLR_RX_DONE)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_RX_DONE:** Read this register to clear the RD_DONE interrupt (bit 7) of the IC_RAW_INTR_STAT register

23.7.21 I2C Clear ACTIVITY Interrupt Register(I2C_CLR_ACTIVITY)

Register	Address offset	Access	Reset value	Description
I2C_CLR_ACTIVITY	0x5C	R	0x0000_0000	I2C Clear ACTIVITY Interrupt Register

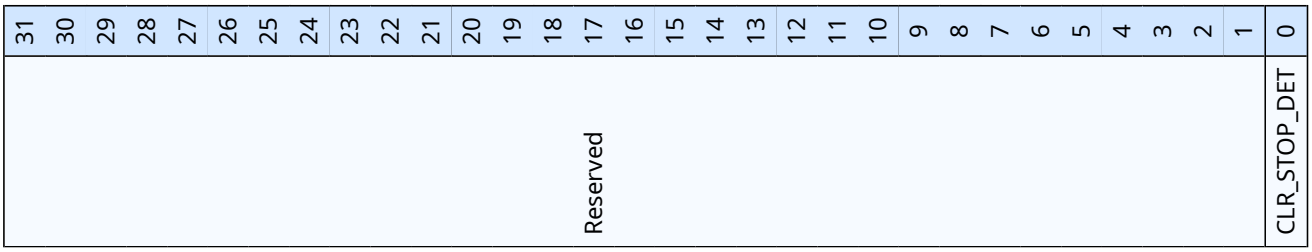


I2C Clear ACTIVITY Interrupt Register(I2C_CLR_ACTIVITY)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_ACTIVITY:** Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the IC_RAW_INTR_STAT register.

23.7.22 I2C Clear STOP_DET Interrupt Register(I2C_CLR_STOP_DET)

Register	Address offset	Access	Reset value	Description
I2C_CLR_STOP_DET	0x60	R	0x0000_0000	I2C Clear STOP_DET Interrupt Register

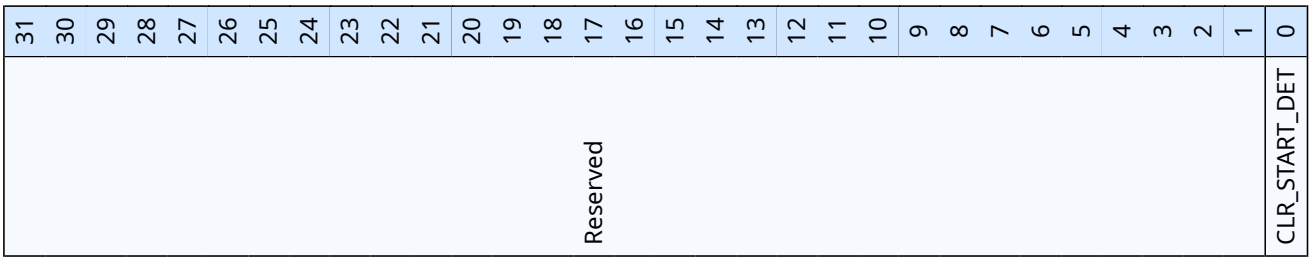


I2C Clear STOP_DET Interrupt Register(I2C_CLR_STOP_DET)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_STOP_DET:** Read this register to clear the STOP_DET interrupt (bit 9) of the IC_RAW_INTR_STAT register.

23.7.23 I2C Clear START_DET Interrupt Register(I2C_CLR_START_DET)

Register	Address offset	Access	Reset value	Description
I2C_CLR_START_DET	0x64	R	0x0000_0000	I2C Clear START_DET Interrupt Register

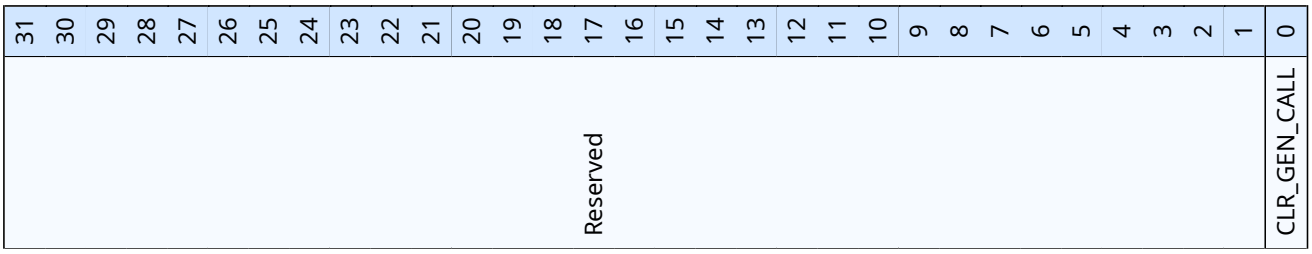


I2C Clear START_DET Interrupt Register(I2C_CLR_START_DET)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_START_DET:** Read this register to clear the START_DET interrupt (bit 10) of the IC_RAW_INTR_STAT register.

23.7.24 I2C Clear GEN_CALL Interrupt Register(I2C_CLR_GEN_CALL)

Register	Address offset	Access	Reset value	Description
I2C_CLR_GEN_CALL	0x68	R	0x0000_0000	I2C Clear GEN_CALL Interrupt Register



I2C Clear GEN_CALL Interrupt Register(I2C_CLR_GEN_CALL)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_GEN_CALL:** Read this register to clear the GEN_CALL interrupt (bit 11) of the IC_RAW_INTR_STAT register.

23.7.25 I2C Enable Register(I2C_ENABLE)

Register	Address offset	Access	Reset value	Description
I2C_ENABLE	0x6C	RW	0x0000_0000	I2C Enable 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
Reserved													SMBUS_ALERT_EN	SMBUS_SUSPEND_EN	SMBUS_CLK_RESET	Reserved												SDA_STUCK_RECOVERY_ENABLE	TX_CMD_BLOCK	ABORT	ENABLE

I2C Enable Register(I2C_ENABLE)bit description

Bit	Access	Description
[31:19]	R	Reserved
[18]	RW	**SMBUS_ALERT_EN:** The SMBUS_ALERT_CTRL register bit is used to control assertion of SMBALERT signal. This register bit is auto-cleared after detection of Acknowledgement from master for Alert Response address. Values: 0x1 (ALERT_ENABLED): Slave initates the Alert signal to indicate SMBus Host 0x0 (SUSPEND_DISABLED): Slave will not initates the Alert signal to indicate SMBus Host.
[17]	RW	**SMBUS_SUSPEND_EN:** The SMBUS_SUSPEND_EN register bit is used to control assertion and de-assertion of SMBSUS signal. Values: 0x1 (ENABLED): Host/Master initates the SMBUS system to enter Suspend Mode. 0x0 (DISABLED): Host/Master will not initates the SMBUS system to enter Suspend Mode.
[16]	RW	**SMBUS_CLK_RESET:** This bit is used in SMBus Host mode to initiate the SMBus Master Clock Reset. This bit should be enabled only when Master is in idle. Whenever this bit is enabled, the SMBCLK is held low for the IC_SCL_STUCK_TIMEOUT ic_clk cycles to reset the SMBus slave devices
[15:4]	R	Reserved
[3]	RW	**SDA_STUCK_RECOVERY_ENABLE:** If SDA is stuck at low indicated through the TX_ABORT interrupt (IC_TX_ABRT_SOURCE[17]), then this bit is used as a control knob to initiate the SDA Recovery Mechanism (that is, send at most 9 SCL clocks and STOP to release the SDA line) and then this bit gets auto clear. Values: 0x1 (SDA_STUCK_RECOVERY_ENABLED): Master initates the SDA stuck at low recovery mechanism. 0x0 (SDA_STUCK_RECOVERY_DISABLED): Master disabled the SDA stuck at low recovery mechanism.
[2]	RW	**TX_CMD_BLOCK:** In Master mode: 1'b1: Blocks the transmission of data on I2C bus even if Tx FIFO has data to transmit. 1'b0: The transmission of data starts on I2C bus automatically, as soon as the first data is available in the Tx FIFO. **Note:** *To block the execution of Master commands, set the TX_CMD_BLOCK bit only when Tx FIFO is empty (IC_STATUS[2]==1) and Master is in Idle state (IC_STATUS[5] == 0). Any further commands put in the Tx FIFO are not executed until TX_CMD_BLOCK bit is unset*

[1]	RW	**ABORT:** When set, the controller initiates the transfer abort. 0: ABORT not initiated or ABORT done 1: ABORT operation in progressThe software can abort the I2C transfer in master mode by setting this bit. The software can set this bit only when ENABLE is already set; otherwise, the controller ignores any write to ABORT bit. The software cannot clear the ABORT bit once set. In response to an ABORT, the controller issues a STOP and flushes the Tx FIFO after completing the current transfer, then sets the TX_ABORT interrupt after the abort operation. The ABORT bit is cleared automatically after the abort operation
[0]	RW	**ENABLE:** 0x1 (ENABLED): I2C is enabled 0x0 (DISABLED): I2C is disabled

23.7.26 I2C Status Register(I2C_STATUS)

Note: *This is a read-only register used to indicate the current transfer status and FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt. When the I2C is disabled by writing 0 in bit 0 of the IC_ENABLE register:*

- *Bits 1 and 2 are set to 1*
- *Bits 3 and 10 are set to 0*

Register	Address offset	Access	Reset value	Description
I2C_STATUS	0x70	R	0x0010_0006	I2C Status 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
Reserved											SMBUS_ALERT_STATUS	SMBUS_SUSPEND_STATUS	SMBUS_SLV_ADDR_RESOLVED	SMBUS_SLV_ADDR_VLD	SMBUS_QUICK_CMD_BIT	Reserved			SAR_DUAL_FLAG	SDA_STUCK_NOT_RECOVERED	SLV_HOLD_RX_FIFO_FULL	SLV_HOLD_TX_FIFO_EMPTY	MST_HOLD_RX_FIFO_FULL	MST_HOLD_TX_FIFO_EMPTY	SLV_ACTIVITY	MST_ACTIVITY	RFF	RFNE	TFE	TFNF	ACTIVITY

I2C Status Register(I2C_STATUS)bit description

Bit	Access	Description
[31:21]	R	Reserved
[20]	R	**SMBUS_ALERT_STATUS:** This bit indicates the status of the SMBus Alert signal (ic_smbalert_in_n). Values: 0x1 (ACTIVE): SMBUS Alert is asserted. 0x0 (INACTIVE): SMBUS Alert is not asserted.
[19]	R	**SMBUS_SUSPEND_STATUS:** This bit indicates the status of the SMBus Suspend signal (ic_smbsus_in_n). Values: 0x1 (ACTIVE): SMBUS System is in Suspended mode. 0x0 (INACTIVE): SMBUS System is not in Suspended mode.

[18]	R	**SMBUS_SLV_ADDR_RESOLVED:** This bit indicates whether the slave address (ic_sar) is resolved by the ARP Master. Values: 0x1 (ACTIVE): SMBUS Slave Address is Resolved. 0x0 (INACTIVE): SMBUS Slave Address is not Resolved.
[17]	R	**SMBUS_SLV_ADDR_VLD:** This bit indicates whether the slave address (ic_sar) is valid or not. Values: 0x1 (ACTIVE): SMBUS Slave Address is Valid. 0x0 (INACTIVE): SMBUS Slave Address is not valid
[16]	RC_R	**SMBUS_QUICK_CMD_BIT:** This bit indicates the R/W bit of the Quick command received. This bit will be cleared after the user has read this bit. Values: 0x1 (ACTIVE): SMBUS QUICK CMD Read/write is set to 1. 0x0 (INACTIVE): SMBUS QUICK CMD Read/write is set to 0.
[15:13]	R	Reserved
[11]	RW	**SAR_DUAL_FLAG:** Double address flag
[11]	RW	**SDA_STUCK_NOT_RECOVERED:** This bit indicates that SDA stuck at low is not recovered after the recovery mechanism. In Slave mode, this register bit is not applicable. Values: 0x1 (ACTIVE): SDA Stuck at low is recovered after recovery mechanism. 0x0 (INACTIVE): SDA Stuck at low is not recovered after recovery mechanism.
[10]	R	**SLV_HOLD_RX_FIFO_FULL:** This bit indicates the BUS Hold in Slave mode due to Rx FIFO is Full and an additional byte has been received (This kind of Bus hold is applicable if IC_RX_FULL_HLD_BUS_EN is set to 1). Values: 0x1 (ACTIVE): Slave holds the bus due to Rx FIFO is full 0x0 (INACTIVE): Slave is not holding the bus or Bus hold is not due to Rx FIFO is full.
[9]	R	**SLV_HOLD_TX_FIFO_EMPTY:** This bit indicates the BUS Hold in Slave mode for the Read request when the Tx FIFO is empty. The Bus is in hold until the Tx FIFO has data to Transmit for the read request. Values: 0x1 (ACTIVE): Slave holds the bus due to Tx FIFO is empty 0x0 (INACTIVE): Slave is not holding the bus or Bus hold is not due to Tx FIFO is empty
[8]	R	**MST_HOLD_RX_FIFO_FULL:** This bit indicates the BUS Hold in Master mode due to Rx FIFO is Full and additional byte has been received (This kind of Bus hold is applicable if IC_RX_FULL_HLD_BUS_EN is set to 1). Values: 0x1 (ACTIVE): Master holds the bus due to Rx FIFO is full 0x0 (INACTIVE): Master is not holding the bus or Bus hold is not due to Rx FIFO is full
[7]	R	**MST_HOLD_TX_FIFO_EMPTY:** The I2C master stalls the write transfer when Tx FIFO is empty, and the last byte does not have the Stop bit set. This bit indicates the BUS hold when the master holds the bus because of the Tx FIFO being empty, and the the previous transferred command does not have the Stop bit set. Values: 0x1 (ACTIVE): Master holds the bus due to Tx FIFO is empty 0x0 (INACTIVE): Master is not holding the bus or Bus hold is not due to Tx FIFO is empty

[6]	R	**SLV_ACTIVITY:** Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set. Values: 0x1 (ACTIVE): Slave not idle 0x0 (IDLE): Slave is idle
[5]	R	**MST_ACTIVITY:** Master FSM Activity Status. When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set. Values: 0x1 (ACTIVE): Master not idle 0x0 (IDLE): Master is idle **Note:** *IC_STATUS[0]-that is, ACTIVITY bit-is the OR of SLV_ACTIVITY and MST_ACTIVITY bits*
[4]	R	**RFF:** Receive FIFO Completely Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. Values: 0x1 (FULL): Rx FIFO is full 0x0 (NOT_FULL): Rx FIFO not full
[3]	R	**RFNE:** Receive FIFO Not Empty. This bit is set when the receive FIFO contains one or more entries; it is cleared when the receive FIFO is empty. Values: 0x1 (NOT_EMPTY): Rx FIFO not empty 0x0 (EMPTY): Rx FIFO is empty
[2]	R	**TFE:** Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. Values: 0x1 (EMPTY): Tx FIFO is empty 0x0 (NON_EMPTY): Tx FIFO not empty
[1]	R	**TFNF:** Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. Values: 0x1 (NOT_FULL): Tx FIFO not full 0x0 (FULL): Tx FIFO is full
[0]	R	**ACTIVITY:** I2C Activity Status. Values: 0x1 (ACTIVE): I2C is active 0x0 (INACTIVE): I2C is idle

23.7.27 I2C Transmit FIFO Level Register(I2C_TXFLR)

Note: *This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever:*

- *The I2C is disabled*
- *There is a transmit abort - that is, TX_ABRT bit is set in the IC_RAW_INTR_STAT register*
- *The slave bulk transmit mode is aborted*
- *The register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.*

Register	Address offset	Access	Reset value	Description
I2C_TXFLR	0x74	R	0x0000_0000	I2C Transmit FIFO Level 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
Reserved																													TXFLR[2:0]		

I2C Transmit FIFO Level Register(I2C_TXFLR)bit description

Bit	Access	Description
[31:3]	R	Reserved
[2:0]	R	**TXFLR[2:0]:** Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

23.7.28 I2C Receive FIFO Level Register(I2C_RXFLR)

Note: *This register contains the number of valid data entries in the receive FIFO buffer. It is cleared whenever:*

- *The I2C is disabled*
- *Whenever there is a transmit abort caused by any of the events tracked in IC_TX_ABRT_SOURCE*

The register increments whenever data is placed into the receive FIFO and decrements when data is taken from the receive FIFO.

Register	Address offset	Access	Reset value	Description
I2C_RXFLR	0x78	R	0x0000_0000	I2C Receive FIFO Level 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
Reserved																													RXFLR[2:0]		

I2C Receive FIFO Level Register(I2C_RXFLR)bit description

Bit	Access	Description
[31:3]	R	Reserved
[2:0]	R	**RXFLR[2:0]:** Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.

23.7.29 I2C SDA Hold Time Length Register(I2C_SDA_HOLD)

Note: *It is only writable when IC_ENABLE[0]=0.*

The values in this register are in units of ic_clk period. The value programmed in IC_SDA_TX_HOLD must be greater than the minimum hold time in each mode one cycle in master mode, seven cycles in slave mode for the value to be implemented.

The programmed SDA hold time during transmit (IC_SDA_TX_HOLD) cannot exceed at any time the duration of the low part of SCL. Therefore the programmed value cannot be larger than N_SCL_LOW-2, where N_SCL_LOW is the duration of the low part of the SCL period measured in ic_clk cycles.

Register	Address offset	Access	Reset value	Description
I2C_SDA_HOLD	0x7C	RW	0x0000_0001	I2C SDA Hold Time Length 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
Reserved								IC_SDA_RX_HOLD								IC_SDA_TX_HOLD[15:0]															

I2C SDA Hold Time Length Register(I2C_SDA_HOLD)bit description

Bit	Access	Description
[31:24]	RW	Reserved
[23:16]	RW	**SDA_RX_HOLD[7:0]:** Sets the required SDA hold time in units of ic_clk period, when the macro acts as a receiver.
[15:0]	RW	**SDA_TX_HOLD[15:0]:** Sets the required SDA hold time in units of ic_clk period, when the macro acts as a transmitter.

23.7.30 I2C Transmit Abort Source Register(I2C_TX_ABRT_SOURCE)

Note: *This register has 32 bits that indicate the source of the TX_ABRT bit. Except for Bit 9, this register is cleared whenever the IC_CLR_TX_ABRT register or the IC_CLR_INTR register is read. To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; RESTART must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]).*

Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, Bit 9 clears for one cycle and is then re-asserted

Register	Address offset	Access	Reset value	Description
I2C_TX_ABRT_SOURCE	0x80	R	0x0000_0000	I2C Transmit Abort Source 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
Reserved						TX_FLUSH_CNT			Reserved					ABRT_SDA_STUCK_AT_LOW	ABRT_USER_ABRT	ABRT_SLVRD_INTX	ABRT_SLV_ARBLOST	ABRT_SLVFLUSH_TXFIFO	ARB_LOST	ABRT_MASTER_DIS	ABRT_10B_RD_NORSTRT	ABRT_SBYTE_NORSTRT	ABRT_HS_NORSTRT	ABRT_SBYTE_ACKDET	ABRT_HS_ACKDET	ABRT_GCALL_READ	ABRT_GCALL_NOACK	ABRT_TXDATA_NOACK	ABRT_10ADDR2_NOACK	ABRT_10ADDR1_NOACK	ABRT_7B_ADDR_NOACK

I2C Transmit Abort Source Register(I2C_TX_ABRT_SOURCE)bit description

Bit	Access	Description
[31:26]	R	Reserved
[25:23]	R	**TX_FLUSH_CNT:** This field indicates the number of Tx FIFO Data Commands which are flushed due to TX_ABRT interrupt. It is cleared whenever I2C is disabled.
[22:18]	R	Reserved
[17]	R	**ABRT_SDA_STUCK_AT_LOW:** This is a master-mode-only bit. Master detects the SDA Stuck at low for the IC_SDA_STUCK_AT_LOW_TIMEOUT value of ic_clks.
[16]	R	**ABRT_USER_ABRT:** This is a master-mode-only bit. Master has detected the transfer abort (IC_ENABLE[1])
[15]	R	**ABRT_SLVRD_INTX:** 0x1 (ABRT_SLVRD_INTX_GENERATED): Slave trying to transmit to remote master in read mode 0x0 (ABRT_SLVRD_INTX_VOID): Slave trying to transmit to remote master in read mode- scenario not present
[14]	R	**ABRT_SLV_ARBLOST:** This field indicates that a Slave has lost the bus while transmitting data to a remote master. IC_TX_ABRT_SOURCE[12] is set at the same time. **Note:** *Even though the slave never 'owns' the bus, something could go wrong on the bus. This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then DW_apb_i2c no longer own the bus*
[13]	R	**ABRT_SLVFLUSH_TXFIFO:** This field specifies that the Slave has received a read command and some data exists in the TX FIFO, so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO.
[12]	R	**ARB_LOST:** This field specifies that the Master has lost arbitration, or if IC_TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration.
[11]	R	**ABRT_MASTER_DIS:** This field indicates that the User tries to initiate a Master operation with the Master mode disabled
[10]	R	**ABRT_10B_RD_NORSTRT:** This field indicates that the restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the master sends a read command in 10-bit addressing mode.
[9]	R	**ABRT_SBYTE_NORSTRT:** 0x1 (ABRT_SBYTE_NORSTRT_GENERATED): User trying to send START byte when RESTART disabled 0x0 (ABRT_SBYTE_NORSTRT_VOID): User trying to send START byte when RESTART disabled- scenario not present
[8]	R	**ABRT_HS_NORSTRT:** This field indicates that the restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the user is trying to use the master to transfer data in High Speed mode.
[7]	R	**ABRT_SBYTE_ACKDET:** This field indicates that the Master has sent a START Byte and the START Byte was acknowledged (wrong behavior)
[6]	R	**ABRT_HS_ACKDET:** This field indicates that the Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior).
[5]	R	**ABRT_GCALL_READ:** This field indicates that DW_apb_i2c in the master mode has sent a General Call but the user programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD[9] is set to 1).
[4]	R	**ABRT_GCALL_NOACK:** This field indicates that DW_apb_i2c in master mode has sent a General Call and no slave on the bus acknowledged the General Call.
[3]	R	**ABRT_TXDATA_NOACK:** This field indicates the master-mode only bit. When the master receives an acknowledgement for the address, but when it sends data byte(s) following the address, it did not receive an acknowledge from the remote slave(s).

[2]	R	**ABRT_10ADDR2_NOACK:** This field indicates that the Master is in 10-bit address mode and that the second address byte of the 10-bit address was not acknowledged by any slave.
[1]	R	**ABRT_10ADDR1_NOACK:** This field indicates that the Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave
[0]	R	**ABRT_7B_ADDR_NOACK:** This field indicates that the Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave.

23.7.31 DMA Control Register(I2C_DMA_CR)

Register	Address offset	Access	Reset value	Description
I2C_DMA_CR	0x88	RW	0x0000_0000	DMA Control 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
Reserved																														TDMAE	RDMAE

DMA Control Register(I2C_DMA_CR)bit description

Bit	Access	Description
[31:2]	R	Reserved
[1]	RW	**TDMAE:** Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel
[0]	RW	**RDMAE:** Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel.

23.7.32 DMA Transmit Data Level Register(I2C_DMA_TDLR)

Register	Address offset	Access	Reset value	Description
I2C_DMA_TDLR	0x8C	RW	0x0000_0000	DMA Transmit Data Level Register

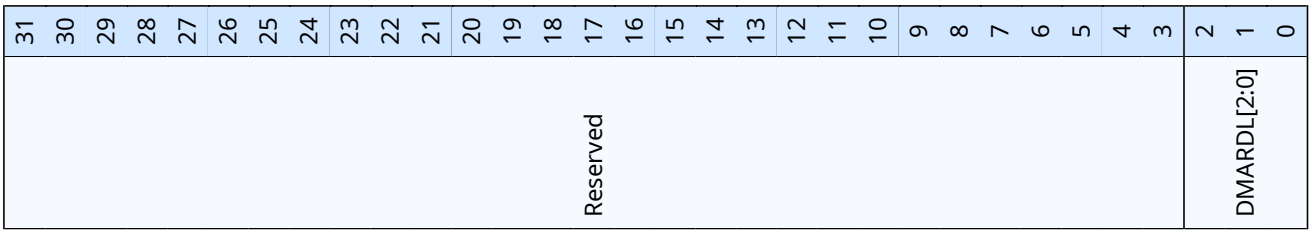
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													DMATDL[2:0]		

DMA Transmit Data Level Register(I2C_DMA_TDLR)bit description

Bit	Access	Description
[31:3]	R	Reserved
[2:0]	RW	**DMATDL[2:0]:** Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.

23.7.33 DMA Receive Data Level Register(I2C_DMA_RDLR)

Register	Address offset	Access	Reset value	Description
I2C_DMA_RDLR	0x90	RW	0x0000_0000	DMA Receive Data Level Register



DMA Receive Data Level Register(I2C_DMA_RDLR)bit description

Bit	Access	Description
[31:3]	R	Reserved
[2:0]	RW	**DMARDL[2:0]:** Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value + 1, and RDMAE =1. For instance, when DMARDL is 0, then dma_rx_req is asserted when 1 or more data entries are present in the receive FIFO.

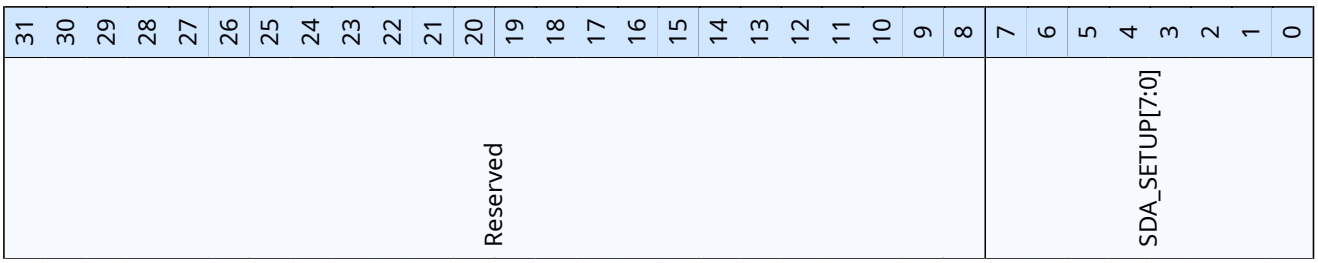
23.7.34 I2C SDA Setup Register(I2C_SDA_SETUP)

Note: *This register controls the amount of time delay (in terms of number of ic_clk clock periods) introduced in the rising edge of SCL - relative to SDA changing - when this macro services a read request in a slave-transmitter operation. This register must be programmed with a value equal to or greater than 2.*

Writes to this register succeed only when IC_ENABLE[0] = 0.

*Note: The length of setup time is calculated using [(IC_SDA_SETUP - 1) * (ic_clk_period)], so if the user requires 10 ic_clk periods of setup time, they should program a value of 11. The IC_SDA_SETUP register is only used when operating as a slave transmitter.*

Register	Address offset	Access	Reset value	Description
I2C_SDA_SETUP	0x94	RW	0x0000_0064	I2C SDA Setup Register



I2C SDA Setup Register(I2C_SDA_SETUP)bit description

Bit	Access	Description
[31:8]	R	Reserved

[7:0]	RW	**SDA_SETUP[7:0]:** SDA Setup. It is recommended that if the required delay is 1000ns, then for an ic_clk frequency of 10 MHz, IC_SDA_SETUP should be programmed to a value of 11. IC_SDA_SETUP must be programmed with a minimum value of 2

23.7.35 I2C ACK General Call Register(I2C_ACK_GENERAL_CALL)

Note: *The register controls whether this macro responds with an ACK or NACK when it receives an I2C General Call address. This register is applicable only when the DW_apb_i2c is in slave mode.*

Register	Address offset	Access	Reset value	Description
I2C_ACK_GENERAL_CALL	0x98	RW	0x0000_0001	I2C ACK General Call Register

I2C ACK General Call Register(I2C_ACK_GENERAL_CALL)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	RW	**ACK_GEN_CALL:** ACK General Call. When set to 1, the macro responds with an ACK when it receives a General Call. Otherwise, the macro responds with a NACK.

23.7.36 I2C Enable Status Register(I2C_ENABLE_STATUS)

Note: *The register is used to report the macro hardware status when the IC_ENABLE[0] register is set from 1 to 0.*
If IC_ENABLE[0] has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1.
If IC_ENABLE[0] has been set to 0, bits 2:1 is only be valid as soon as bit 0 is read as '0'.
When IC_ENABLE[0] has been set to 0, a delay occurs for bit 0 to be read as 0 because disabling the macro depends on I2C bus activities.

Register	Address offset	Access	Reset value	Description
I2C_ENABLE_STATUS	0x9C	R	0x0000_0000	I2C Enable Status Register

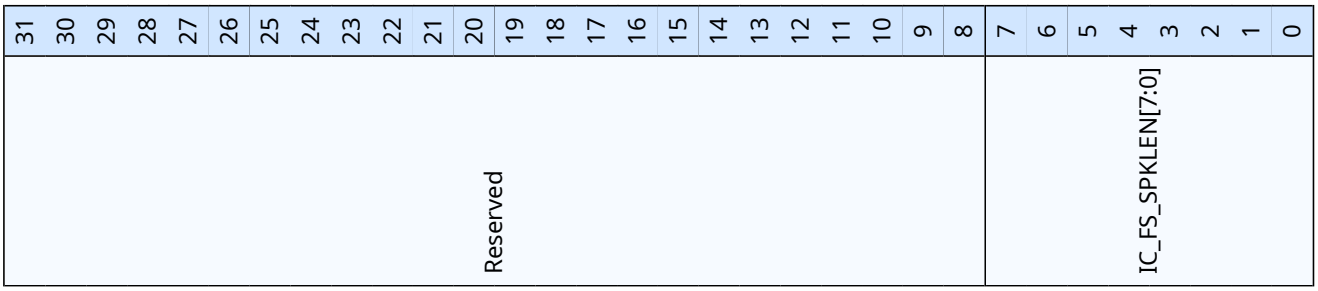
I2C Enable Status Register(I2C_ENABLE_STATUS)bit description

Bit	Access	Description
[31:1]	R	Reserved
[2]	R	**SLV_RX_DATA_LOST:** Slave Received Data Lost. Values: 0x1 (ACTIVE): Slave RX Data is lost 0x0 (INACTIVE): Slave RX Data is not lost

[1]	R	**SLV_DISABLED_WHILE_BUSY:** Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to the setting bit 0 of the IC_ENABLE register from 1 to 0. Values: 0x1 (ACTIVE): Slave is disabled when it is active 0x0 (INACTIVE): Slave is disabled when it is idle
[0]	R	**IC_EN:** 0x1 (ENABLED): I2C enabled 0x0 (DISABLED): I2C disabled

23.7.37 I2C SS, FS spike suppression limit(I2C_FS_SPKLEN)

Note: *This register is used to store the duration, measured in ic_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in SS, FS modes.*
This register can be written only when IC_ENABLE[0]=0



Register	Address offset	Access	I2C Reset value	Description
I2C_FS_SPKLEN	0xA0	RW	0x0000_0001	I2C SS, FS spike suppression limit

I2C SS, FS spike suppression limit(I2C_FS_SPKLEN)bit description

Bit	Access	Description
[31:8]	R	Reserved
[7:0]	RW	**IC_FS_SPKLEN:** This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that will be filtered out by the spike suppression logic.

23.7.38 I2C SCL Stuck at Low Timeout register(I2C_SCL_STUCK_AT_LOW_TIMEOUT)

Note: *This register can be written only when IC_ENABLE[0]=0*

Register	Address offset	Access	Reset value	Description
I2C_SCL_STUCK_AT_LOW_TIMEOUT	0xAC	RW	0xFFFF_FFFF	I2C SCL Stuck at Low Timeout register

I2C SCL Stuck at Low Timeout register(I2C_SCL_STUCK_AT_LOW_TIMEOUT)bit description

Bit	Access	Description
[31:0]	RW	**SCL_STUCK_AT_LOW_TIMEOUT:** This register is used to store the duration, measured in ic_clk cycles, used to Generate an Interrupt (SCL_STUCK_AT_LOW) if SCL is held low for the IC_SCL_STUCK_LOW_TIMEOUT duration.

23.7.39 I2C SDA Stuck at Low Timeout register(I2C_SDA_STUCK_AT_LOW_TIMEOUT)

Note: *This register can be written only when IC_ENABLE[0]=0*

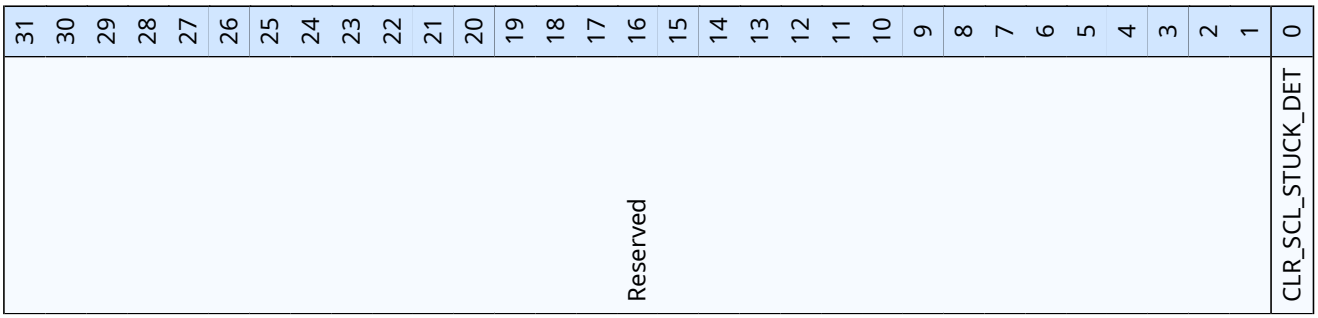
Register	Address offset	Access	Reset value	Description
I2C_SDA_STUCK_AT_LOW_TIMEOUT	0xB0	RW	0xFFFF_FFFF	I2C SDA Stuck at Low Timeout register

SCL SDA Stuck at Low Timeout register(I2C_SCL_STUCK_AT_LOW_TIMEOUT)bit description

Bit	Access	Description
[31:0]	RW	**SCL_STUCK_AT_LOW_TIMEOUT:** This register is used to store the duration, measured in ic_clk cycles, used to Recover the Data (SDA) line through sending SCL pulses if SDA is held low for the mentioned duration.

23.7.40 I2C Clear SCL Stuck at Low Detect interrupt Register(I2C_CLR_SCL_STUCK_DET)

Register	Address offset	Access	Reset value	Description
I2C_CLR_SCL_STUCK_DET	0xB4	R	0x0000_0000	I2C Clear SCL Stuck at Low Detect interrupt Register



I2C Clear SCL Stuck at Low Detect interrupt Register(I2C_CLR_SCL_STUCK_DET)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**CLR_SCL_STUCK_DET:** Read this register to clear the SCL_STUCT_AT_LOW interrupt (bit 15) of the IC_RAW_INTR_STAT register.

23.7.41 SMBus Slave Clock Extend Timeout register(SMBUS_CLK_LOW_SEXT)

Note: *This Register contains the Timeout value used to determine the Slave Clock Extend Timeout in one transfer (from START to STOP).*

This register can be written only when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn't exist in I2C2.

Register	Address offset	Access	Reset value	Description
SMBUS_CLK_LOW_SEXT	0xBC	RW	0xFFFF_FFFF	SMBus Slave Clock Extend Timeout register

SMBus Slave Clock Extend Timeout register(SMBUS_CLK_LOW_SEXT)bit description

Bit	Access	Description

[31:0]	RW	**SMBUS_CLK_LOW_SEXT_TIMEOUT:** This field is used to detect the Slave Clock Extend time-out ($t_{LOW:SEXT}$) in master mode extended by the slave device in one message from the initial START to the STOP. The values in this register are in units of ic_clk period.

23.7.42 SMBus Master Clock Extend Timeout register(SMBUS_CLK_LOW_MEXT)

Note: *This Register contains the Timeout value used to determine the Master Clock Extend Timeout in one byte of transfer. This register can be written only when IC_ENABLE[0]=0. This register only exists in I2C1, and doesn't exist in I2C2.*

Register	Address offset	Access	Reset value	Description
SMBUS_CLK_LOW_MEXT	0xC0	RW	0xFFFF_FFFF	SMBus Master Clock Extend Timeout register

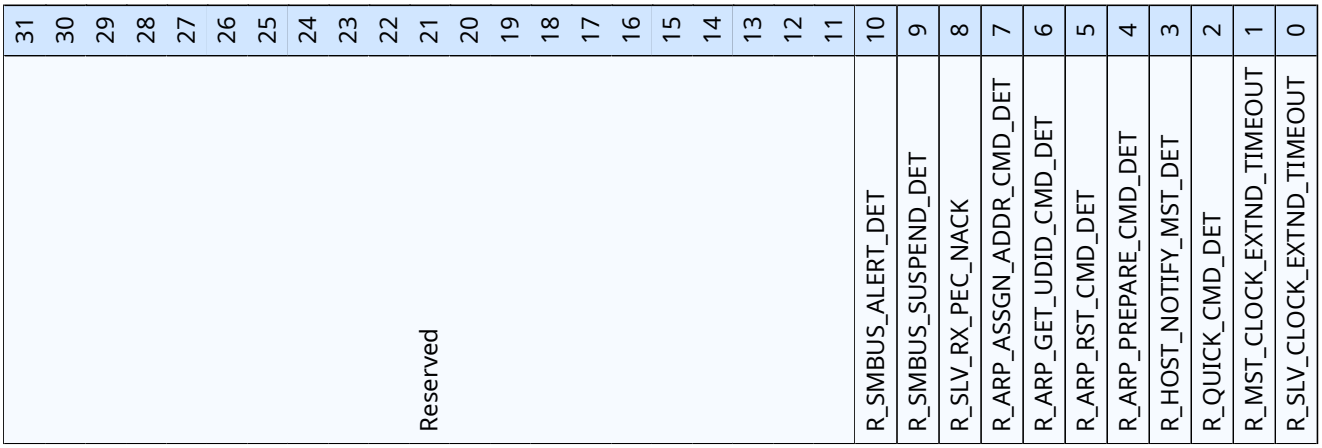
SMBus Master Clock Extend Timeout register(SMBUS_CLK_LOW_MEXT)bit description

Bit	Access	Description
[31:0]	RW	**SMBUS_CLK_LOW_MEXT_TIMEOUT:** This field is used to detect the Master extend SMBus clock (SCLK) timeout defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP in Master mode. The values in this register are in units of ic_clk period.

23.7.43 SMBus Interrupt Status Register(SMBUS_INTR_STAT)

Note: *Each bit in this register has a corresponding mask bit in the IC_SMBUS_INTR_MASK register. These bits are cleared by writing the matching SMBus interrupt clear register(IC_CLR_SMBUS_INTR) bits. The unmasked raw versions of these bits are available in the IC_SMBUS_RAW_INTR_STAT register. The interrupt status bits are high active.*
This register only exists in I2C1, and doesn't exist in I2C2

Register	Address offset	Access	I2C1 Reset value	Description
SMBUS_INTR_STAT	0xC8	R	0x0000_0000	SMBus Interrupt Status Register



SMBus Interrupt Status Register(SMBUS_INTR_STAT)bit description

Bit	Access	Description
[31:11]	R	Reserved
[10]	R	**R_SMBUS_ALERT_DET:** Status for interrupt SMBUS_ALERT_DET
[9]	R	**R_SMBUS_SUSPEND_DET:** Status for SMBUS_SUSPEND_DET
[8]	R	**R_SLV_RX_PEC_NACK:** Status for SLV_RX_PEC_NACK

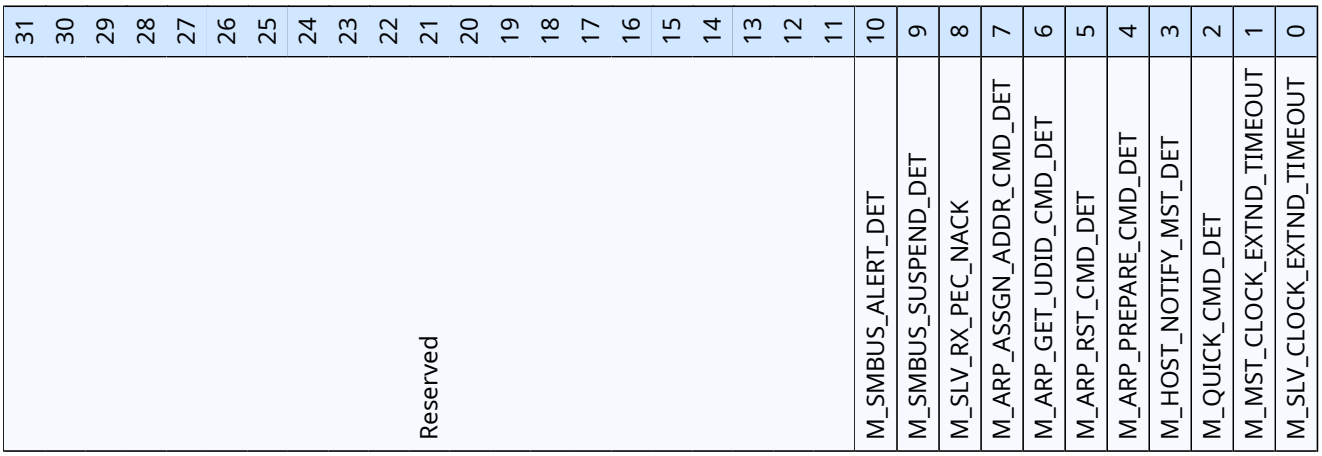
[7]	R	**R_ARP_ASSGN_ADDR_CMD_DET:** Status for ARP_ASSGN_ADDR_CMD_DET
[6]	R	**R_ARP_GET_UDID_CMD_DET:** Status for ARP_GET_UDID_CMD_DET
[5]	R	**R_ARP_RST_CMD_DET:** Status for ARP_RST_CMD_DET
[4]	R	**R_ARP_PREPARE_CMD_DET:** Status for ARP_PREPARE_CMD_DET
[3]	R	**R_HOST_NOTIFY_MST_DET:** Status for HOST_NOTIFY_MST_DET
[2]	R	**R_QUICK_CMD_DET:** Status for QUICK_CMD_DET
[1]	R	**R_MST_CLOCK_EXTND_TIMEOUT:** Status for MST_CLOCK_EXTND_TIMEOUT
[0]	R	**R_SLV_CLOCK_EXTND_TIMEOUT:** Status for SLV_CLOCK_EXTND_TIMEOUT

23.7.44 SMBus Interrupt Mask Register(SMBUS_INTR_MASK)

Note: *These bits mask their corresponding interrupt status bits. This register is active low; a value of 0 masks the interrupt, whereas a value of 1 unmasks the interrupt.*

This register only exists in I2C1, and doesn't exist in I2C2.

Register	Address offset	Access	I2C1 Reset value	Description
SMBUS_INTR_MASK	0xCC	R	0x0000_07FF	SMBus Interrupt Mask Register



SMBus Interrupt Mask Register(I2C_INTR_MASK)bit description

Bit	Access	Description
[31:11]	R	Reserved
[10]	R	**M_SMBUS_ALERT_DET:** Mask bit for SMBUS_ALERT_DET
[9]	R	**M_SMBUS_SUSPEND_DET:** Mask bit for SMBUS_SUSPEND_DET
[8]	R	**M_SLV_RX_PEC_NACK:** Mask bit for SLV_RX_PEC_NACK
[7]	R	**M_ARP_ASSGN_ADDR_CMD_DET:** Mask bit for ARP_ASSGN_ADDR_CMD_DET
[6]	R	**M_ARP_GET_UDID_CMD_DET:** Mask bit for ARP_GET_UDID_CMD_DET
[5]	R	**M_ARP_RST_CMD_DET:** Mask bit for ARP_RST_CMD_DET
[4]	R	**M_ARP_PREPARE_CMD_DET:** Mask bit for ARP_PREPARE_CMD_DET
[3]	R	**M_HOST_NOTIFY_MST_DET:** Mask bit for HOST_NOTIFY_MST_DET
[2]	R	**M_QUICK_CMD_DET:** Mask bit for QUICK_CMD_DET
[1]	R	**M_MST_CLOCK_EXTND_TIMEOUT** Mask bit for MST_CLOCK_EXTND_TIMEOUT
[0]	R	**M_SLV_CLOCK_EXTND_TIMEOUT:** Mask bit for SLV_CLOCK_EXTND_TIMEOUT

23.7.45 SMBus Raw Interrupt Status Register(SMBUS_RAW_INTR_STAT)

Note: *This register only exists in I2C1, and doesn't exist in I2C2.*

Register	Address offset	Access	I2C1 Reset value	Description
SMBUS_RAW_INTR_STAT	0xD0	R	0x0000_0000	SMBus Raw Interrupt Status 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
Reserved																					SMBUS_ALERT_DET	SMBUS_SUSPEND_DET	SLV_RX_PEC_NACK	ARP_ASSGN_ADDR_CMD_DET	ARP_GET_UDID_CMD_DET	ARP_RST_CMD_DET	ARP_PREPARE_CMD_DET	HOST_NOTIFY_MST_DET	QUICK_CMD_DET	MST_CLOCK_EXTND_TIMEOUT	SLV_CLOCK_EXTND_TIMEOUT

SMBus Raw Interrupt Status Register(SMBUS_RAW_INTR_STAT)bit description

Bit	Access	Description
[31:11]	R	Reserved
[10]	R	**SMBUS_ALERT_DET:** Indicates whether a SMBALERT (ic_smbalert_in_n) signal is driven low by the slave.
[9]	R	**SMBUS_SUSPEND_DET:** Indicates whether a SMBSUS (ic_smbsus_in_n) signal is driven low by the Host
[8]	R	**SLV_RX_PEC_NACK:** Indicates whether a NACK has been sent due to PEC mismatch while working as ARP slave.
[7]	R	**ARP_ASSGN_ADDR_CMD_DET:** Indicates whether an Assign Address ARP command has been received
[6]	R	**GET_UDID_CMD_DET:** Indicates whether a Get UDID ARP command has been received.
[5]	R	**ARP_RST_CMD_DET:** Indicates whether a General or Directed Reset ARP command has been received
[4]	R	**ARP_PREPARE_CMD_DET:** Indicates whether a prepare to ARP command has been received.
[3]	R	**HOST_NTFY_MST_DET:** Indicates whether a Notify ARP Master ARP command has been received.
[2]	R	**QUICK_CMD_DET:** Indicates whether a Quick command has been received on the SMBus interface regardless of whether this macro is operating in slave or master mode. Enabled only when IC_SMBUS=1 is set to 1.
[1]	R	**MST_CLOCK_EXTND_TIMEOUT:** Indicates whether the Master device transaction (START-to-ACK, ACK-to-ACK, or ACK-to-STOP) from START to STOP exceeds IC_SMBUS_CLOCK_LOW_MEXT time with in each byte of message. This bit is enabled only when: • IC_SMBUS=1 • IC_CON[0]=1 • IC_EMPTYFIFO_HOLD_MASTER_EN=1 or IC_RX_FULL_HLD_BUS_EN=1

[0]	R	**SLV_CLOCK_EXTND_TIMEOUT:** Indicates whether the transaction from Slave (i.e from START to STOP) exceeds IC_SMBUS_CLK_LOW_SEXT time. This bit is enabled only when: IC_SMBUS=1 IC_CON[0]=1

23.7.46 Clear SMBus Interrupt Register(CLR_SMBUS_INTR)

Register	Address offset	Access	I2C1 Reset value	Description
CLR_SMBUS_INTR	0xD4	R	0x0000_0000	Clear SMBus Interrupt 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
Reserved																					CLR_SMBUS_ALERT_DET	CLR_SMBUS_SUSPEND_DET	CLR_SLV_RX_PEC_NACK	CLR_ARP_ASSGN_ADDR_CMD_DET	CLR_ARP_GET_UDID_CMD_DET	CLR_ARP_RST_CMD_DET	CLR_ARP_PREPARE_CMD_DET	CLR_HOST_NOTIFY_MST_DET	CLR_QUICK_CMD_DET	CLR_MST_CLOCK_EXTND_TIMEOUT	CLR_SLV_CLOCK_EXTND_TIMEOUT

Clear SMBus Interrupt Register(CLR_SMBUS_INTR)bit description

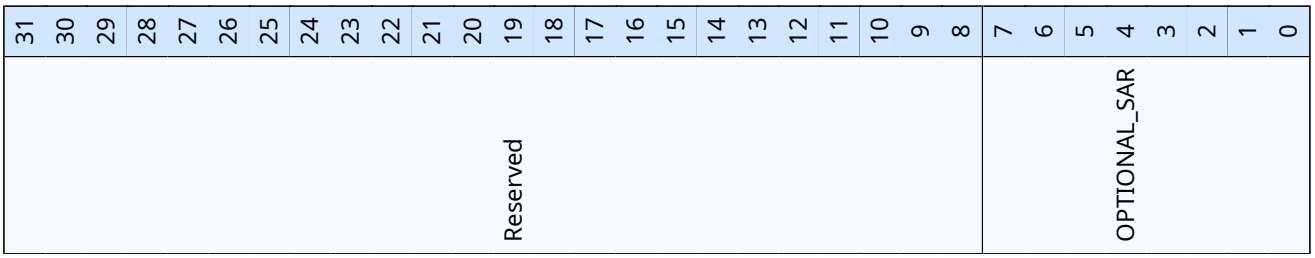
Bit	Access	Description
[31:11]	R	Reserved
[10]	W	**CLR_SMBUS_ALERT_DET:** Clear bit for SMBUS_ALERT_DET
[9]	W	**CLR_SMBUS_SUSPEND_DET:** Clear bit for SMBUS_SUSPEND_DET
[8]	W	**CLR_SLV_RX_PEC_NACK:** Clear bit for SLV_RX_PEC_NACK
[7]	W	**CLR_ARP_ASSGN_ADDR_CMD_DET:** Clear bit for ARP_ASSGN_ADDR_CMD_DET
[6]	W	**CLR_GET_UDID_CMD_DET:** Clear bit for ARP_GET_UDID_CMD_DET
[5]	W	**CLR_RST_CMD_DET:** Clear bit for ARP_RST_CMD_DET
[4]	W	**CLR_ARP_PREPARE_CMD_DET:** Clear bit for ARP_PREPARE_CMD_DET
[3]	W	**CLR_HOST_NTFY_MST_DET:** Clear bit for HOST_NOTIFY_MST_DET
[2]	W	**CLR_QUICK_CMD_DET:** Clear bit for QUICK_CMD_DET
[1]	W	**CLR_CLOCK_EXTND_TIMEOUT:** Clear bit for MST_CLOCK_EXTND_TIMEOUT
[0]	W	**CLR_SLV_CLOCK_EXTND_TIMEOUT:** Clear bit for SLV_CLOCK_EXTND_TIMEOUT

23.7.47 I2C Optional Slave Address Register(I2C_OPTIONAL_SAR)

Note:

1. *This register can be written only when IC_ENABLE[0]=0.*
2. *This register only exists in I2C1, and doesn't exist in I2C2.*

Register	Address offset	Access	Reset value	Description
I2C_OPTIONAL_SAR	0xD8	RW	0x0000_0000	I2C Optional Slave Address Register



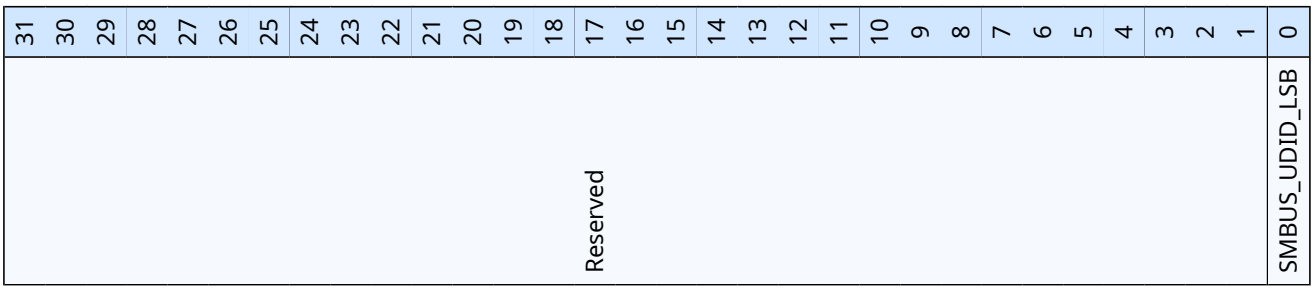
I2C Optional Slave Address Register(I2C_OPTIONAL_SAR)bit description

Bit	Access	Description
[31:8]	R	Reserved
[7:0]	RW	**OPTIONAL_SAR[7:0]:** Optional Slave address for DW_apb_i2c when operating as a slave in SMBus Mode.

23.7.48 SMBUS ARP UDID LSB Register(SMBUS_UDID_LSB)

Note:

1. *This register can be written only when IC_ENABLE[0]=0.*
2. *This register only exists in I2C1, and doesn't exist in I2C2.*



Register	Address offset	Access	Reset value	Description
SMBUS_UDID_LSB	0xDC	RW	0xFFFF_FFFF	SMBUS ARP UDID LSB Register

SMBUS ARP UDID LSB Register(SMBUS_UDID_LSB)bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**SMBUS_UDID_LSB:** This field is used to store the LSB 32 bit value of slave unique device identifier used in Address Resolution Protocol.

24 Secure digital input/output interface (SDIO)

24.1 SDIO main features

The SDIO controller provides an interface between the host computer (AHB) and MMC cards, SD memory cards, SDIO devices and CE-ATA devices.

- Fully compatible with Multimedia Card Version 4.2 protocol. Supports three different data bus modes: 1-bit, 4-bit and 8-bit.
- Fully compatible with SD memory card version 2.0 protocol.
- Compatible with SDIO version 2.0 protocol: two different data bus modes are supported: 1-bit and 4-bit.
- Not compatible with the SPI mode specified in the SDIO protocol.
- Fully compatible with CE-ATA version 1.1

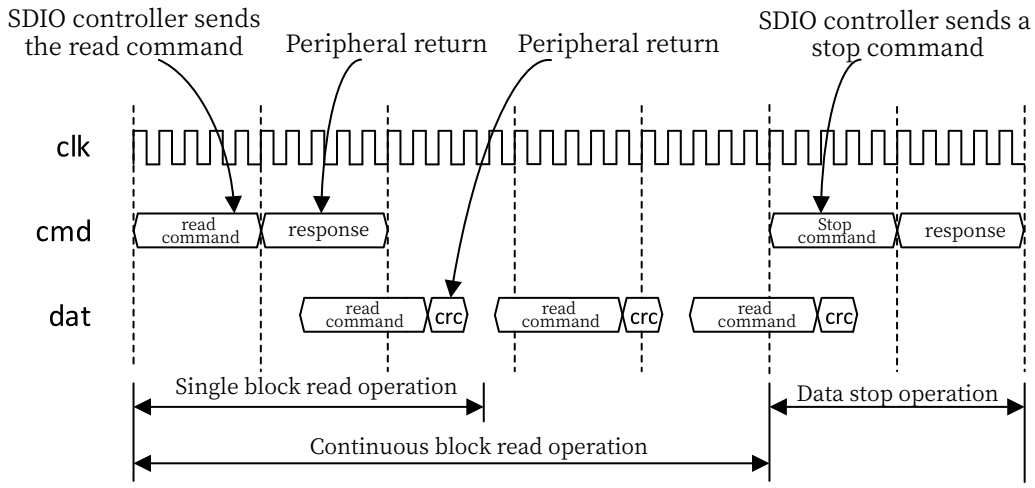
24.2 SDIO Bus

The SDIO bus consists of 3 signals:

- CLK (clock) line, unidirectional, generated by the SDIO controller and sent to the peripheral;
- CMD (command) line, bidirectional, which is sent by the SDIO controller before the peripheral returns the response information;
- DATA (data) line, bidirectional, can be divided DATA (data) line, bi-directional, can complete the write operation (SDIO controller -> peripheral) and read operation (peripheral -> SDIO controller) in time;

The following is a schematic diagram of consecutive multi-block read operations, first the SDIO controller sends a read command to obtain a response, and at the same time starts to obtain data from the peripheral A new stop command is sent after enough data is acquired to end the read operation.

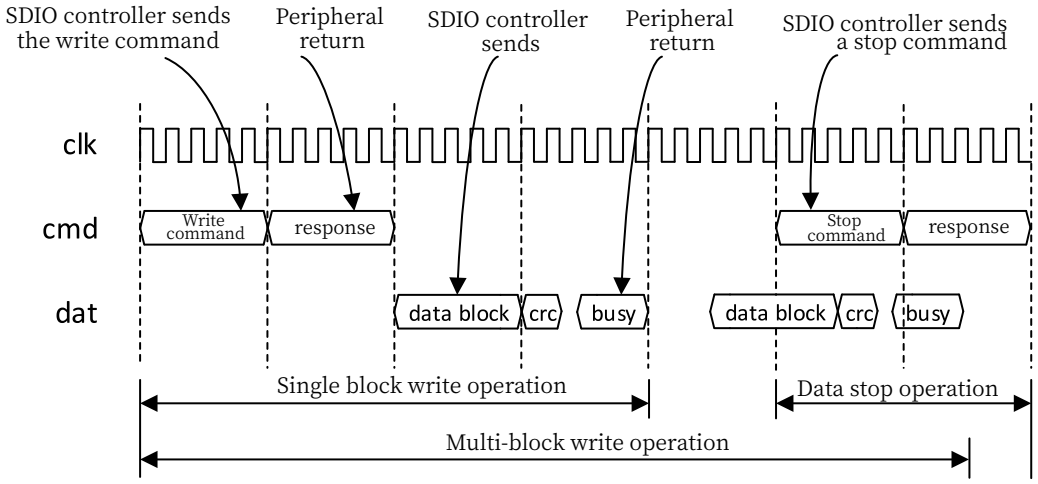
Fig 24.2-1 Multi-block read operation



Successive write operations are similar, but after each write, you have to check if the peripheral is completed, and when it is not completed, BUSY is high, and then start the next write operation after

BUSY is pulled low. The next write operation is started after BUSY is pulled low.

Fig 24.2-2 Multi-block write operations

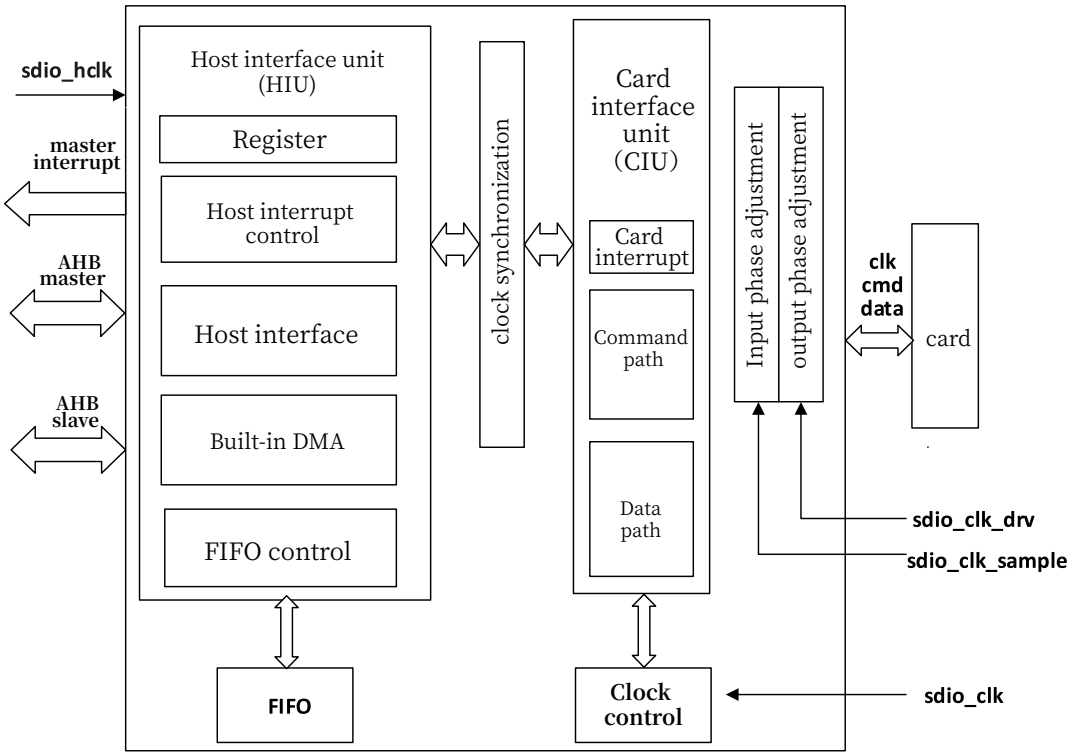


24.3 Controller Description

The SDIO controller mainly consists of

- HIU, the host interface unit, operates on the system SDIO_HCLK and is responsible for handling host-side signals: receiving host commands from the AHB slave, sending DMA move signals from the AHB master, and generating host interrupts. The HIU, the Host Interface Unit, works in the system SDIO_HCLK and is responsible for handling host-side signals: receiving host commands from the AHB slave, sending DMA move signals from the AHB master, generating host interrupts.
- CIU, Card Interface Unit, works on SDIO_CLK and is responsible for handling card-side signals: receiving card responses/read data/card interrupts, etc., sending commands/writing data, etc. Write data, etc.
- The signals between HIU and CIU must be clock synchronized

Fig 24.3-1 SDIO Controller Block Diagram



24.3.1 Host Interface Unit (HIU)

The host controls the SDIO controller through an AHB, through which the internal registers can be read and written.

The SDIO controller eventually generates an interrupt to the host, which is sent by multiple interrupt sources ”or” together after independent enable switches. After receiving the interrupt, the interrupt source can be acknowledged through two sets of interrupt observation registers.

The SDIO controller contains a built-in DMA that can independently perform data transfer between the FIFO and the system MEM, which can be done automatically through descriptor configuration. The descriptor is also stored in the system MEM independently and the built-in DMA will get it automatically.

The SDIO controller has an internal 512-byte FIFO to store read and write data, which is automatically updated by the hardware through control registers and card bus status. The hardware is automatically updated through the control registers and the card bus status.

24.3.2 Card Interface Unit (CIU)

The Card Interface Unit is responsible for completing the connection between the HIU and the card bus, including the processing of commands, responses, data and the use and updating of some control and status registers. to ensure that the signals entering the card bus fully meet the protocol requirements.

24.4 SDIO Register

Tab 24.4-1 SDIO registers map

offset	Register	Reset value	Description
SDIO_BA = 0x4001_8000			
0x00	CTRL	0x0000_0000	SDIO control register
0x08	CLKDIV	0x0000_0000	SDIO Clock division configuration
0x10	CLKENA	0x0000_0000	SDIO Clock enable register
0x14	TMOUT	0xFFFF_FF40	Timeout counter configuration
0x18	CTYPE	0x0000_0000	Card Type Register
0x1C	BKSIZ	0x0000_0020	Block size configuration
0x20	BYTCNT	0x0000_0200	Byte counter
0x24	INTMASK	0x0000_0000	Interrupt mask configuration
0x28	CMDARG	0x0000_0000	Command parameter configuration
0x2C	CMD	0x0000_0000	Command configuration
0x30	RESP0	0x0000_0000	Response 0
0x34	RESP1	0x0000_0000	Response 1
0x38	RESP2	0x0000_0000	Response 2
0x40	MINTSTS	0x0000_0000	Mask interrupt information
0x44	RINTSTS	0x0000_0000	Unmasked interrupt information
0x48	STATUS	0x0000_0000	Status information
0x4C	FIFOTH	0x0000_0000	FIFO threshold configuration
0x54	WRTPRT	0x0000_0000	Write protection configuration
0x60	TBBCNT	0x0000_0000	BIU sends completion byte counter
0x80	BMOD	0x0000_0000	Bus mode configuration
0x84	PLDMND	0x0000_0000	Rotation training requirements for suspension recovery
0x88	DBADDR	0x0000_0000	Descriptor base address
0x8C	IDSTS	0x0000_0000	DMA status
0x90	IDINTEN	0x0000_0000	DMA interrupt enabled
0x94	DSCADDR	0x0000_0000	Current descriptor address
0x98	BUFADDR	0x0000_0000	Current BUF address
0x100	CardThrCtl	0x0000_0000	The card threshold is set

24.4.1 CTRL

Register	Address offset	Access	Reset value	Description
CTRL	0x00	RW	0x0000_0000	SDIO control 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
Reserved						use_internal_dmac	Reserved													ceata_device_interrupt_status	send_auto_stop_ccsd	send_ccsd	abort_read_data	send_irq_response	read_wait	dma_enable	int_enable	Reserved	dma_reset	fifo_reset	controller_reset

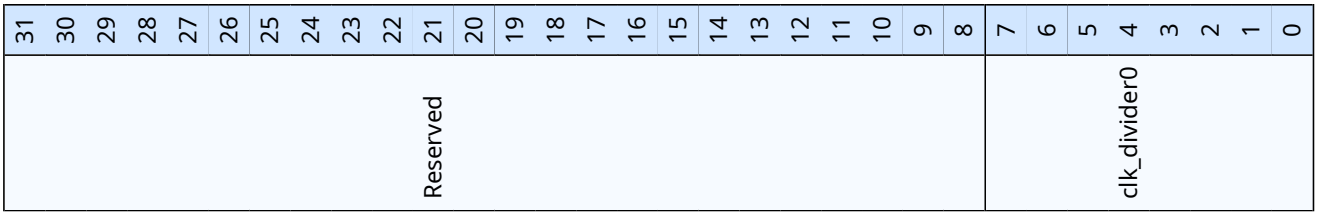
SDIO control register(CTRL)bit description

Bit	Access	Description
[31:26]	R	Reserved
[25]	RW	**use_internal_dmac:** FIFO read/write control options: 0: Read and write by master; 1: Read and write by DMA
[24: 12]	R	Reserved
[11]	RW	**ceata_device_interrupt_status:** 0: CE-ATA device is not interrupt enabled 1: CE-ATA device has interrupts enabled;
[10]	RW	**send_auto_stop_ccsd:** Needs to be set with send_ccsd, set to 1 After sending the CCSD to the CE-ATA device Then the system automatically sends a STOP command The hardware will automatically reset the bit and produce it at the same time ACD production is interrupted
[9]	RW	**send_ccsd:** Write 1: After two card clock cycles, the CCSD command is sent to the CE-ATA device. After the CCSD command is sent, the hardware automatically clears the CCSD bit and generates a CD interrupt
[8]	RW	**abort_read_data:** After the suspending command is issued for read transmission, the software needs to poll the status of the card to confirm the completion of the suspending operation. After the confirmation, the bit needs to be written to 1 to restart the sending state machine to prevent the state machine from continuing to wait for the next data. After the restart, the hardware will automatically clear the bit to zero. Generally, it is used to suspend the SDIO card.
[7]	RW	**send_irq_response:** Generally, after CMD40 is sent to the MMC card, it needs to wait for the interrupt response of the card. If the bit is written as 1 when sending, this wait can be skipped. The state machine directly enters the idle state after sending and automatically clears the bit to zero.
[6]	RW	**read_wait:** Write 1 send read - Wait to SDIO card, write 0 clear
[5]	RW	**dma_enable:** DMA enable; 0: disables. 1: Open;
[4]	RW	**int_enable:** Interrupt is always enabled. 0: disables. 1: Open
[3]	R	Reserved
[2]	RW	**dma_reset:** DMA reset enable; 0: unchanged; 1: reset;
[1]	RW	**fifo_reset:** FIFO reset enable; 0: unchanged; 1: reset;

[0]	RW	**controller_reset:** Controller reset enable; 0: unchanged; 1: reset;

24.4.2 CLKDIV

Register	Address offset	Access	Reset value	Description
CLKDIV	0x08	RW	0x0000_0000	SDIO Clock division configuration

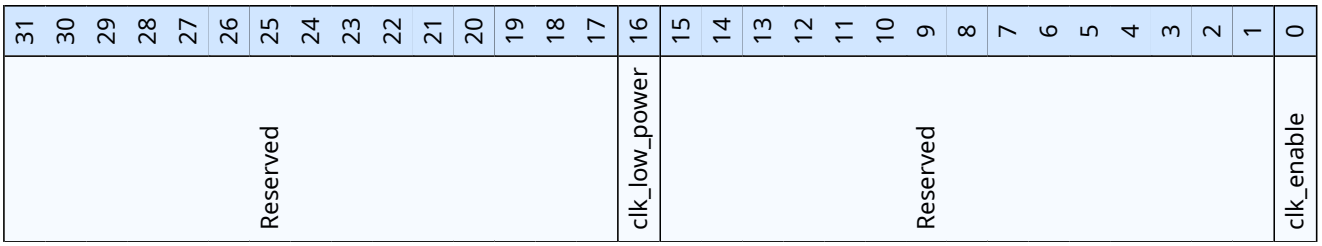


CLKDIV Register bit description

Bit	Access	Description
[31:8]	R	Reserved
[7:0]	RW	**clk_divider0:** Clock divider 0:1 division (clock unchanged); 1:2 divider; 2:4 divisions; 3:6 divisions; 4:8 divisions; ... 255:510 divisions

24.4.3 CLKENA

Register	Address offset	Access	Reset value	Description
CLKENA	0x10	RW	0x0000_0000	SDIO Clock enable register

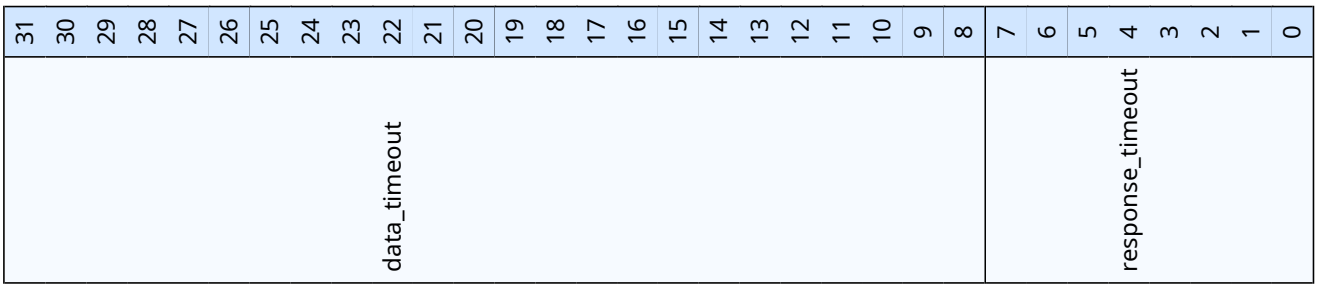


CLKENA register bit description

Bit	Access	Description
[31:17]	R	Reserved
[16]	RW	**clk_low_power:** Low-power clock enable 0: off; 1: stop sending when in vacant state
[15:1]	R	Reserved
[0]	RW	**clk_enable:** Card clock enable 0: close; 1: Open;

24.4.4 TMOUT

Register	Address offset	Access	Reset value	Description
TMOUT	0x14	RW	0xFFFF_FF40	Timeout counter configuration

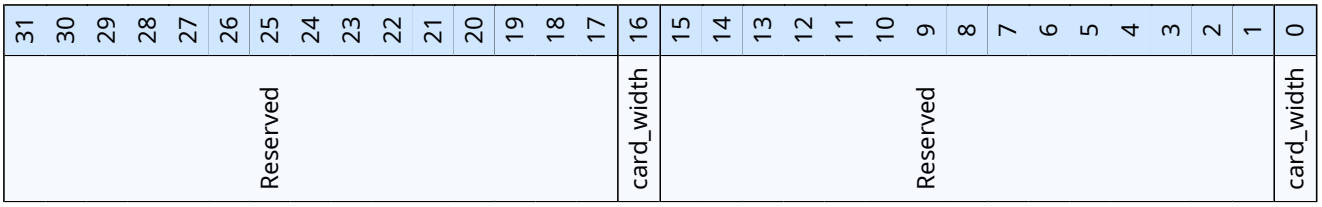


TMOUT register bit description

Bit	Access	Description
[31:8]	RW	**data_timeout:** Card data read timeout threshold; The timeout counter starts after the card clock stops and then sets sdio_clk starts counting; It is best to use an external counter if the timeout threshold exceeds 100ms;
[7:0]	RW	**clk_enable:** Response timeout threshold; Wait for the response to start counting sdio_clk;

24.4.5 CTYPE

Register	Address offset	Access	Reset value	Description
CTYPE	0x18	RW	0x0000_0000	Card Type Register



CTYPE register bit description

Bit	Access	Description
[31:17]	R	Reserved
[16]	RW	**card_width:** Indicates whether the card is 8-bit: 1: Non-8-bit mode 0: 8-bit mode
[15:1]	R	Reserved
[0]	RW	**card_width:** Bit16 0 is used to indicate whether the card is 1 bit or 4 bit: 0:1 bit mode

24.4.6 BKSIZ

Register	Address offset	Access	Reset value	Description
BKSIZ	0x1C	RW	0x0000_0020	Block size configuration

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
Reserved																block_size															

BKSIZ register bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:0]	RW	**block_size:** Block size

24.4.7 BYTCNT

Register	Address offset	Access	Reset value	Description
BYTCNT	0x20	RW	0x0000_0200	Byte counter

BYTCNT register bit description

Bit	Access	Description
[31:0]	RW	**byte_count:** The number of bytes to be transferred must be an integer multiple of the size of the block transferred. For an undefined number of bytes transferred, the byte count should be set to 0, at which point the software sends a stop/abort command to terminate the data transfer

24.4.8 INTMASK

Register	Address offset	Access	Reset value	Description
INTMASK	0x24	RW	0x0000_0000	Interrupt mask configuration

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
Reserved															sdio_int_mask	EBE mask	ACD mask	SBE/BCI mask	HLE mask	FRUN mask	HTO mask	DRTO mask	RTO mask	DCRC mask	RCRC mask	RXDR mask	TXDR mask	DTO mask	CD mask	RE mask	Reserved

INTMASK register bit description

Bit	Access	Description
[31:17]	R	Reserved
[16]	RW	**sdio_int_mask:** SDIO interrupt enable, 0: off; 1: on;
[15]	RW	**EBE mask:** Write operation end bit error interrupt or read operation no CRC interrupt enable , 0: off; 1: on;
[14]	RW	**ACD mask:** Automatic command completion interrupt enable, 0: off; 1: ON
[13]	RW	**SBE/BCI mask:** A start bit error or BUSY bit clears the interrupt enable. 0: off; 1: on;
[12]	RW	**HLE mask:** Hardware miswrite operation interrupt enable, 0: off; 1: Open
[11]	RW	**FRUN mask:** FIFO overflow interrupt enable, 0: off; 1: on on
[10]	RW	**HTO mask:** Data missing timeout interrupt enable, 0: off; 1: ON
[9]	RW	**DRTO mask:** Read data timeout interrupt enable, 0: off; 1: Open

[8]	RW	**RTO mask:** Response timeout interrupt enable, 0: off; 1: on on
[7]	RW	**DCRC mask:** Data CRC error interrupt enable, 0: off; 1: ON
[6]	RW	**RCRC mask:** Enable in response to CRC error interrupt, 0: off; 1: ON
[5]	RW	**RXDR mask:** Receive FIFO data request interrupt enable, 0: off closed; 1: open;
[4]	RW	**TXDR mask:** Send FIFO data request interrupt enable, 0: off Closed; 1: Open
[3]	RW	**DTO mask:** Data transfer completion interrupt enable, 0: off; 1: Open
[2]	RW	**CD mask:** Command completion interrupt enable, 0: off; 1: on on
[1]	RW	**RE mask:** Response error interrupt enable, 0: off; 1: on on;
[0]	R	Reserved

24.4.9 CMDARG

Register	Address offset	Access	Reset value	Description
CMDARG	0x28	RW	0x0000_0000	Command parameter configuration

CMDARG register bit description

Bit	Access	Description
[31:0]	RW	**cmd_arg:** Command parameters passed to the card

24.4.10 CMD

Register	Address offset	Access	Reset value	Description
CMD	0x2C	RW	0x0000_0000	Command configuration

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
start_cmd	reserved	use_hold_reg	volt_switch	boot_mode	disable_boot	expect_boot_ack	enable_boot	ccs_expected	read_ceata_device	update_clock_registers_only	reserved	send_initialization	stop_abort_cmd	wait_prvdata_complete	send_auto_stop	transfer_mode	read/write	data_expected	check_response_crc	response_length	response_expect	cmd_index

CMD register bit description

Bit	Access	Description
[31]	RW	**start_cmd:** Write 1 to start the command. After the CIU receives the command, the hardware automatically clears the command. Software does not write any more command registers until clear. Once written, an HLE (hardware miswrite) interrupt is triggered
[31]	RW	**cmd_arg:** Command parameters passed to the card
[31:0]	R	Reserved
[29]	RW	**use_hold_reg:** 0: CMD and DATA are sent directly to the card 1: CMD and DATA are sent to the card after passing the HOLD register
[26]	RW	**volt_switch:** 0: no voltage switching 1: Voltage switching enabled; must be set only for CMD11

[25]	RW	**expect_boot_ack:** Boot confirmation selection. When the software sets this bit in conjunction with enable_boot, the CIU requires a 0-1-0 response from the selected card to confirm entering boot mode
[24]	RW	**enable_boot:** Enable Boot - Effective only when boot mode is forced. When the software sets this bit with start_cmd, CIU starts the boot sequence of the corresponding card by setting the CMD line to low. Do not set disable_boot and enable_boot at the same time.
[23]	RW	**ccs_expected:** 0: Interrupt is not enabled on the CE-ATA device, or the command does not require the CCS from the device. 1: Interrupt is enabled on the CE-ATA device, and the RW_BLK command requires a command completion signal from the CE-ATA device
[22]	RW	**read_ceata_device:** 0: The master has not performed a read access to the CE-ATA device (RW_REG or RW_BLK); 1: The master is performing a read access to the CE-ATA device
[21]	RW	**update_clock_registers_only:** 0: Normal command sequence 1: No command is sent, only the clock register is updated (CLKDIV/CLKENA) values to the card clock field
[20:16]	RW	reserved
[15]	RW	**send_initialization:** 0: The initialization sequence is not sent before the command is sent. 1: The initialization sequence (1 of 80 card clocks) is sent before the command is sent. Generally, the sequence needs to be inserted when the first command is sent to the card.
[14]	RW	**stop_abort_cmd:** 1: The stop or abort command is designed to stop the current data transfer in progress.
[13]	RW	**wait_prvdata_complete:** 0: Send the command immediately, even if the previous data transfer has not completed 1: Wait for the previous data transfer to complete before sending the command
[12]	RW	**send_auto_stop:** 0: The stop command is not sent when data transmission is complete 1: sends a stop command when data transmission is complete
[11]	RW	**transfer_mode:** 0 : Block data transfer command 1 : Stream data transfer command
[10]	RW	**read/write:** 0: Read the card; 1: Writing the card
[9]	RW	**data_expected:** 0: no data transfer (read/write) 1: with data transfer (read/write)
[8]	RW	**check_response_crc:** 0: No response CRC check 1: Check response CRC
[7]	RW	**response_length:** 0 :Short response ; 1 :Long response
[6]	RW	**response_expect:** 0 :No card response 1 :With card response
[5:0]	RW	**cmd_index:** command Index

24.4.11 RESP0

Register	Address offset	Access	Reset value	Description
RESP0	0x30	R	0x0000_0000	Response 0

RESP0 register bit description

Bit	Access	Description
[31:0]	R	**RESP0:** Long and short response bits 31- bits 0

24.4.12 RESP1

Register	Address offset	Access	Reset value	Description
RESP1	0x34	R	0x0000_0000	Response 1

RESP1 register bit description

Bit	Access	Description
[31:0]	R	**response1:** Long response bit 63 - bit 32

24.4.13 RESP2

Register	Address offset	Access	Reset value	Description
RESP2	0x38	R	0x0000_0000	Response 2

RESP1 register bit description

Bit	Access	Description
[31:0]	R	**response2:** Long response bit 127 - bit 96

24.4.14 MINTSTS

Register	Address offset	Access	Reset value	Description
MINTSTS	0x40	R	0x0000_0000	Mask interrupt information

MINTSTS register bit description

Bit	Access	Description
[31:17]	R	Reserved
[16]	R	**sdio_int:** SDIO interrupt state, affected by INTMASK
[15]	R	**EBE:** Write operation end bit error interrupt status or read operation No CRC interrupt status, affected by INTMASK
[14]	R	**ACD:** Automatic command completion interrupt status,affected by INTMASK
[13]	R	**SBE/BCI:** Start bit error or BUSY bit clears interrupt status, affected by INTMASK
[12]	R	**HLE:** hardware mis-write operation interrupt status, affected by INTMASK
[11]	R	**FRUN:** FIFO overflow interrupt status, affected by INTMASK
[10]	R	**HTO:** Data missing timeout interrupt status, affected by INTMASK

[9]	R	**DRTO:** Read data timeout interrupt status, affected by INTMASK
[8]	R	**RTO:** Response timeout interrupt status, affected by INTMASK
[7]	R	**DCRC:** Data CRC error interrupt status, affected by INTMASK
[6]	R	**RCRC:** Enable in response to CRC error interrupt status, affected by INTMASK
[5]	R	**RXDR:** Receive FIFO data request interrupt status, affected by INTMASK
[4]	R	**TXDR:** Send FIFO data request interrupt status, affected by INTMASK
[3]	R	**DTO:** Data transfer completion interrupt status, affected by INTMASK
[2]	R	**CD:** Command completion interrupt status, affected by INTMASK
[1]	R	**RE:** Response error interrupt status, affected by INTMASK
[0]	R	Reserved

24.4.15 RINTSTS

Register	Address offset	Access	Reset value	Description
RINTSTS	0x44	R	0x0000_0000	Unmasked interrupt information

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
Reserved															sdio_int	EBE	ACD	SBE/BCI	HLE	FRUN	HTO	DRTO	RTO	DCRC	RCRC	RXDR	TXDR	DTO	CD	RE	Reserved

RINTSTS register bit description

Bit	Access	Description
[31:17]	R	Reserved
[16]	R	**sdio_int:** SDIO interrupt state, Unaffected by INTMASK
[15]	R	**EBE:** Write operation end bit error interrupt status or read operation No CRC interrupt status, Unaffected by INTMASK
[14]	R	**ACD:** Automatic command completion interrupt status,Unaffected by INTMASK
[13]	R	**SBE/BCI:** Start bit error or BUSY bit clears interrupt status, Unaffected by INTMASK
[12]	R	**HLE:** hardware mis-write operation interrupt status, Unaffected by INTMASK
[11]	R	**FRUN:** FIFO overflow interrupt status, Unaffected by INTMASK
[10]	R	**HTO:** Data missing timeout interrupt status, Unaffected by INTMASK
[9]	R	**DRTO:** Read data timeout interrupt status, Unaffected by INTMASK
[8]	R	**RTO:** Response timeout interrupt status, Unaffected by INTMASK
[7]	R	**DCRC:** Data CRC error interrupt status, Unaffected by INTMASK
[6]	R	**RCRC:** Enable in response to CRC error interrupt status, Unaffected by INTMASK
[5]	R	**RXDR:** Receive FIFO data request interrupt status, Unaffected by INTMASK
[4]	R	**TXDR:** Send FIFO data request interrupt status, Unaffected by INTMASK
[3]	R	**DTO:** Data transfer completion interrupt status, Unaffected by INTMASK
[2]	R	**CD:** Command completion interrupt status, Unaffected by INTMASK
[1]	R	**RE:** Response error interrupt status, Unaffected by INTMASK
[0]	R	Reserved

24.4.16 STATUS

Register	Address offset	Access	Reset value	Description
STATUS	0x48	R	0x0000_0000	Status information

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
dma_req	dma_ack	Reserved													response_index						data_state_mc_busy	data_busy	data_3_status	command fsm states				fifo_full	fifo_empty	fifo_tx_watermark	fifo_rx_watermark

STATUS register bit description

Bit	Access	Description
[31]	R	**dma_req:** DMA request signal status
[30]	R	**dma_ack:** DMA Acknowledgement of signal status
[29:17]	R	**fifo_count:** FIFO Counting - FIFO Filled Number of locations
[16:11]	R	**response_index:** Index value of the previous response
[10]	R	**data_state_mc_busy:** Data sending or receiving status machine busy
[9]	R	**data_busy:** By making card_data[0] busy Detection result. 0: card data is not busy 1: Card data is busy
[8]	R	**data_3_status:** By detecting the existence of card_data[3] card. 0: The card does not exist 1: The card exists
[7]	R	**command fsm states:** Command FSM status: 0: idle 1: sends the initialization sequence 2: indicates the start bit of Tx cmd 3: Tx cmd indicates the tx bit 4: Tx cmd index + arg 5 : Tx cmd crc7 6: Tx cmd end bit 7: Rx response start bit 8: Rx response IRQ response 9: Rx responds to the tx bit 10: Rx responds to cmd idx 11: Rx response data 12: Rx responds to crc7 13: Rx response end bit 14: Cmd path waits for NCC 15: Wait, CMD to the response process
[3]	R	**fifo_full:** FIFO is full
[2]	R	**fifo_empty:** FIFO is empty
[1]	R	**fifo_tx_watermark:** FIFO less than or equal to the send threshold
[0]	R	**fifo_rx_watermark:** FIFO is greater than the reception threshold

24.4.17 FIFOTH

Register	Address offset	Access	Reset value	Description
FIFOTH	0x4C	RW	0x0000_0000	FIFO threshold configuration

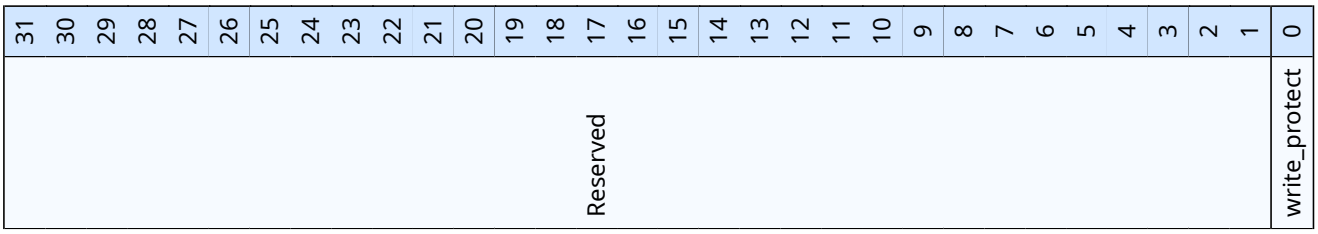
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DMA_Multiple_Transaction_Size			RX_WMark												Reserved				TX_WMark											

FIFOTH register bit description

Bit	Access	Description
[31]	RW	Reserved
[30:28]	RW	**DMA_Multiple_Transaction_Size:** Length of data per DMA transfer (in 32 bits) 000 –1 001 –4 010 –8 011 –16 100 –32 101 –64 110 –128 111 –256
[27:16]	RW	**RX_WMark:** FIFO threshold for receiving card data. When the number of data in the FIFO is greater than this value: in DMA mode, DMA request is triggered; In non-DMA mode, an interrupt is generated if a FIFO threshold interrupt is enabled
[15:12]	R	Reserved
[11:0]	RW	**TX_WMark:** FIFO threshold for sending card data. When the number of data in the FIFO is less than or equal to this value: In DMA mode, a DMA request is triggered. In non-DMA mode, an interrupt is generated if a FIFO threshold interrupt is enabled.

24.4.18 WRTPRT

Register	Address offset	Access	Reset value	Description
WRTPRT	0x54	R	0x0000_0000	Write protection configuration



WRTPRT register bit description

Bit	Access	Description
[31:1]	R	Reserved
[0]	R	**write_protect:** Status of the card write protection port: 1: indicates write protection

24.4.19 TCBCNT

Register	Address offset	Access	Reset value	Description
TCBCNT	0x5C	R	0x0000_0000	CIU sends completion byte counter

TCBCNT register bit description

Bit	Access	Description
[31:0]	R	**trans_card_byte_count:** Number of bytes transferred from CIU to card

24.4.20 TBBCNT

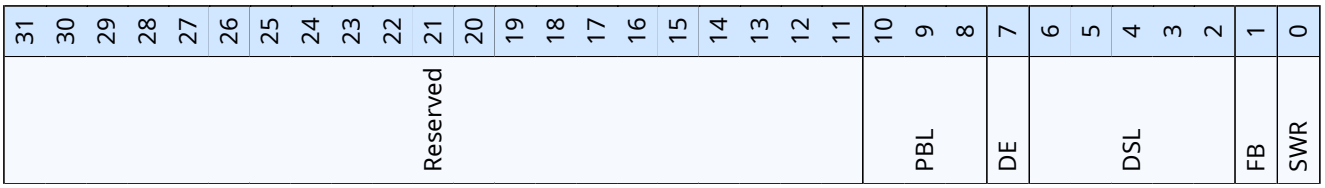
Register	Address offset	Access	Reset value	Description
TBBCNT	0x60	R	0x0000_0000	BIU sends completion byte counter

TBBCNT register bit description

Bit	Access	Description
[31:0]	R	**rans_fifo_byte_count:** Number of bytes transferred between software (or DMA) and HIU FIFO

24.4.21 BMOD

Register	Address offset	Access	Reset value	Description
BMOD	0x80	RW	0x0000_0000	Bus mode configuration



BMOD register bit description

Bit	Access	Description
[31:11]	R	Reserved
[10:8]	R	**PBL:** The DMA_Multiple_Transaction_Size value configured in FIFOTH

[7]	RW	**DE:** Built-in DMA enable switch 0: off 1: on
[6:2]	R	**DSL:** Descriptor jump length, that is, the number of intervals between two non-linked descriptors. Only for dual BUF mode.
[1]	R	**FB:** Fixed Burst. When set to 1, it is transmitted in a fixed burst manner
[0]	R	**SWR:** Soft reset. When set to 1, the DMA controller resets all internal registers, after which the bit is self-cleared to zero

24.4.22 PLDMND

Register	Address offset	Access	Reset value	Description
PLDMND	0x84	W	0x0000_0000	Rotation training requirements for suspension recovery

PLDMND register bit description

Bit	Access	Description
[31:0]	R	**PD:** Polling requirement register. If the OWN bit of the descriptor is not set, FSM enters the suspended state. The host needs to write any value to PD to resume the fetch operation of the built-in DMA.

24.4.23 DBADDR

Register	Address offset	Access	Reset value	Description
DBADDR	0x88	RW	0x0000_0000	Descriptor base address

DBADDR register bit description

Bit	Access	Description
[31:0]	RW	**SDL:** The starting address of the descriptor list, bit1-bit0, must be fixed to 0.

24.4.24 IDSTS

Register	Address offset	Access	Reset value	Description
IDSTS	0x8C	RW	0x0000_0000	DMA status

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
Reserved															FSM				FBE_CODE			AIS	NIS	Reserved		CES	DU	Reserved	FBE	RI	TI

IDSTS register bit description

Bit	Access	Description
[31:17]	R	Reserved

[16:13]	R	**FSM:** Current status of built-in DMA FSM: 0 –DMA_IDLE 1 –DMA_SUSPEND 2 –DESC_RD 3 –DESC_CHK 4 –DMA_RD_REQ_WAIT 5 –DMA_WR_REQ_WAIT 6 –DMA_RD 7 –DMA_WR 8 –DESC_CLOSE
[12:10]	R	**FBE_CODE:** Bus access error and abort: 001 - Abort while sending 010 - Abort on receiving Other - Reserved
[9]	RW	**AIS:** Summary of abnormal interrupt. Is the logic of IDSTS[2] and IDSTS[4] or, write 1 to clear 0
[8]	RW	**NIS:** Normal interrupt summary. Is the logic of IDSTS[0] and IDSTS[1] or, write 1 to clear 0
[7:6]	R	Reserved
[5]	RW	**CES:** Summary of card errors. Is the logic or of RINTST's following bits: • EBE • RTO • RCRC • SBE • DRTO • DCRC • RE Write the 1 to clear the 0
[4]	RW	**DU:** Descriptor acquisition failure interrupt status. The OWN bit is set to 0. Write the 1 to clear the 0
[3]	R	Reserved
[2]	RW	**FBE:** Bus access error interrupted state. Write the 1 to clear the 0
[1]	RW	**RI:** Receive interrupt. Indicates that a descriptor of data reception is complete, write 1 to clear 0
[0]	RW	**TI:** Send interrupt. Indicates that a descriptor of data sent is complete, write 1 to clear 0

24.4.25 IDINTEN

Register	Address offset	Access	Reset value	Description
IDINTEN	0x90	RW	0x0000_0000	DMA interrupt enabled

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
Reserved																						AI mask	NI mask	Reserved		CES mask	DU mask	Reserved	FBE mask	RI mask	TI mask

IDINTEN register bit description

Bit	Access	Description
[31:10]	R	Reserved
[9]	RW	**AIS:** The abnormal interrupt summary function was enabled. Enable IDSTS[2] and IDSTS[4]
[8]	RW	**NIS:** Normal interrupt summary was enabled. Enable IDSTS[0] and IDSTS[1]
[7:6]	R	Reserved
[5]	RW	**CES mask:** The card error summary interrupt function was enabled
[4]	RW	**DU mask:** Description failed to obtain interrupt enable
[3]	R	Reserved
[2]	RW	**FBE mask:** Fatal bus error interrupt enable
[1]	RW	**RI mask:** Receive interrupt enable
[0]	RW	**TI mask:** Send interrupt enable

24.4.26 DSCADDR

Register	Address offset	Access	Reset value	Description
DSCADDR	0x94	R	0x0000_0000	Current descriptor address

DSCADDR register bit description

Bit	Access	Description
[31: 0]	R	**HDA:** The start address of the descriptor read by the current built-in DMA.

24.4.27 BUFADDR

Register	Address offset	Access	Reset value	Description
BUFADDR	0x98	R	0x0000_0000	Current BUF address

BUFADDR register bit description

Bit	Access	Description
[31: 0]	R	**HBA:** Current BUF start address for data read by built-in DMA

24.4.28 CardThrCtl

Register	Address offset	Access	Reset value	Description
CardThrCtl	0x100	RW	0x0000_0000	The card threshold is set

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
Reserved						CardThreshold										Reserved													CardWrThrEn	BsyClrIntEn	CardRdThrEn

CardThrCtl register bit description

Bit	Access	Description
[31:26]	R	Reserved
[25:16]	RW	**CardThreshold:** Card threshold
[15:3]	R	Reserved
[2]	RW	**CardWrThrEn:** 1: Open the write threshold of the card to start, that is, TX FIFO data greater than the card threshold to start to write data transmission
[1]	RW	**BsyClrIntEn:** 0: BUSY clears interrupt closed 1: BUSY clears interrupt enabled
[0]	RW	**CardRdThrEn:** 1: Open the card read threshold start, that is, the space in RX FIFO is greater than the card threshold to start the read data transmission

25 USB

25.1 USB introduction

The Universal serial bus full-speed device interface (USB) module implements the functionality of a USB2.0 full-speed peripheral and consists of a USB controller and a USB physical layer transceiver

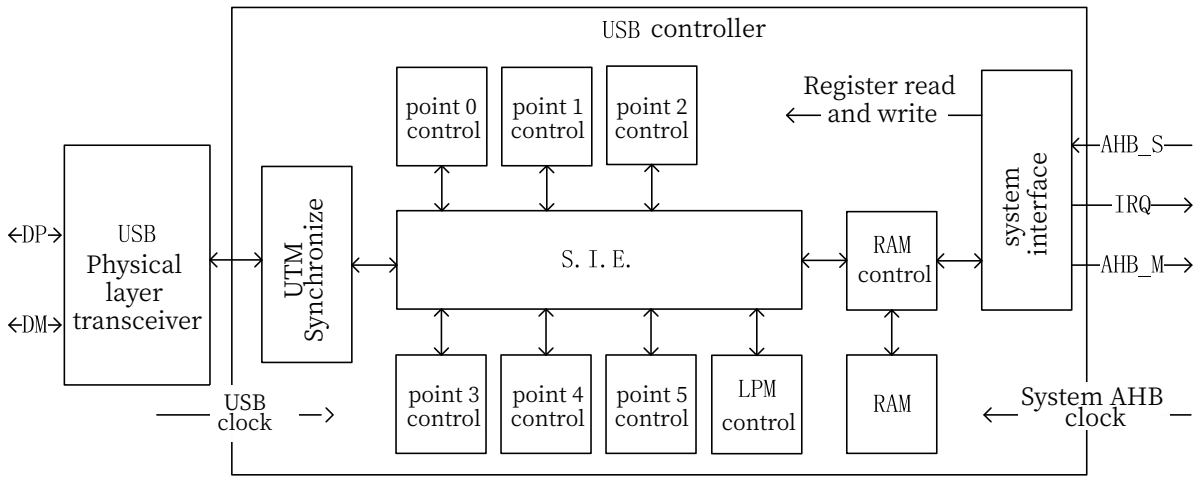
25.2 USB main features

- Comply with USB2.0 full speed device specification
- Can be configured with 1 to 6 endpoints
- All endpoints dynamically allocate FIFO
- The FIFO contains a maximum of 1024 bytes
- Supports dual-buffered batch/synchronous transfers
- Support for control/interrupt transmission
- Supports USB suspend/restore operations
- 4 built-in DMA channels
- Soft connection/disconnection is supported

25.3 USB block diagram

AHB_S acts as the system control bus and is used to configure and access the registers of the USB controller.
AHB_M serves as USB's built-in DMA bus, which is used to access external memory space

Fig 25.3-1 Multi-block read operation



25.4 Clock configuration

The USB controller uses two clocks: the USB clock and the system AHB clock.

- USB Clock:
  - The controller requires a fixed 60MHz clock, which is generated internally by the USB physical layer transceiver.

- System AHB clock:
  - Refer to RCMU to open EN.
  - Note that the system AHB clock frequency can be configured. If the frequency is too low, data buffer overflow may occur

25.4.1 Endpoint control

The USB controller contains six endpoint controllers.
Each endpoint controller contains a TX controller that handles IN transactions (send) and an RX controller that handles OUT transactions (receive), and a FIFO for each TX and RX controller:

- All 16 Fifos together share 1024 bytes of RAM.
- Endpoint 0 has a fixed FIFO depth of 64 bytes and can only buffer one packet.
- The FIFO depth of other endpoints can be configured in 8-256 bytes, buffering one or two packets.

25.5 USB Register

Tab 25.5-1 USB registers map

offset	Register	Reset value
USB_BA = 0x4000_5C00		
0x00	FADDR	0x0000_0000
0x01	POWER	0x0000_0000
0x02	IntrTx	0x0000_0000
0x04	IntrRx	0x0000_0000
0x06	IntrTxE	0x0000_003F
0x08	IntrRxE	0x0000_003E
0x0A	IntrUSB	0x0000_0000
0x0B	IntrUSBE	0x0000_0006
0x0C	FRAME	0x0000_0000
0x0E	INDEX	0x0000_0000
0x1N0/0x10	TxMaxP(Endpoints 1-5)	0x0000_0000
0x102/0x12	CSR0LH(Endpoints 0)	0x0000_0000
0x1N2/0x12	TXCSRLH(Endpoints 1-5)	0x0000_0000
0x1N4/0x14	RxMaxP(Endpoints 1-5)	0x0000_0000
0x1N6/0x16	RXCSRLH(Endpoints 1-5)	0x0000_0000
0x108/0x18	Count0(Endpoints 0)	0x0000_0000
00x1N8/0x18	RxCount(Endpoints 1-5)	0x0000_0000
0x1NF/0x1F	FIFOSize(Endpoints 1-5)	0x0000_0000
0x20	FIFO0 (Endpoints 0)	0x0000_0000
0x24	FIFO1(Endpoints 1)	0x0000_0000
0x28	FIFO2(Endpoints 2)	0x0000_0000
0x2C	FIFO3(Endpoints 3)	0x0000_0000
0x30	FIFO4(Endpoints 4)	0x0000_0000
0x34	FIFO5(Endpoints 5)	0x0000_0000
0x61	MISC	0x0000_0000
0x62	TxFIFOsz(Endpoints 1-5)	0x0000_0000
0x63	RxFIFOsz(Endpoints 1-5)	0x0000_0000
0x64	TxFIFOadd(Endpoints 0-5)	0x0000_0000
0x66	RxFIFOadd(Endpoints 0-5)	0x0000_0000
0x7a	LinkInfo	0x5C
0x7d	FS_EOF1	0x77
0x7f	SOFT_RST	0x0000_0000
0x200	DMA_INTR	0x0000_0000
0x204	CH1_CNTL	0x0000_0000
0x208	CH1_ADDR	0x0000_0000
0x20C	CH1_COUNT	0x0000_0000

(continue)

offset	Register	Reset value
0x214	CH2_CNTL	0x0000_0000
0x218	CH2_ADDR	0x0000_0000
0x21C	CH2_COUNT	0x0000_0000
0x224	CH3_CNTL	0x0000_0000
0x228	CH3_ADDR	0x0000_0000
0x22C	CH3_COUNT	0x0000_0000
0x234	CH4_CNTL	0x0000_0000
0x238	CH4_ADDR	0x0000_0000
0x23C	CH4_COUNT	0x0000_0000
0x340	RxDPktBufDis	0x0000_0000
0x342	TxDPktBufDis	0x0000_0000
0x360	LPM_ATTR	0x0000_0000
0x362	LPM_CNRL	0x0000_0000
0x363	LPM_INTERN	0x0000_0000
0x364	LPM_INTR	0x0000_0000

25.5.1 FADDR

Register	Address offset	Access	Reset value
FADDR	0x00	RW	0x0000_0000

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
Reserved																									Func Addr						

FADDR register bit description

Bit	Access	Description
[31:7]	R	Reserved
[6:0]	RW	**Func Addr:** The external address is allocated by the host computer, sent by the SET_ADDRESS command, and then written by the CPU

25.5.2 POWER

Register	Address offset	Access	Reset value
POWER	0x01	RW	0x0000_0000

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
Reserved																								ISO Update	Soft Conn	HS Enab	HS Mode	Reset	Resume	Suspend Mode	Reserved

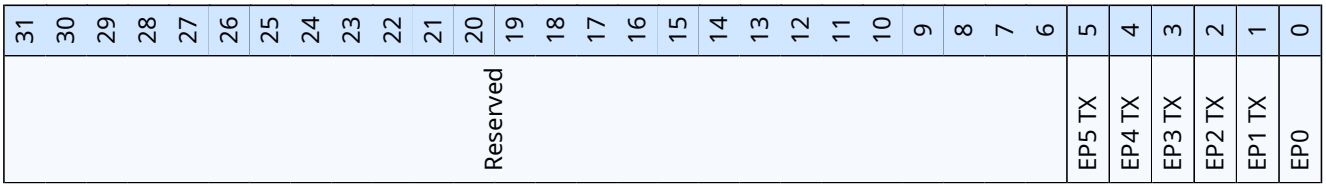
POWER register bit description

Bit	Access	Description
[31:8]	R	Reserved
[7]	RW	**ISO Update:** Synchronous transmission update mode, valid for all endpoints performing synchronous transmission. Write 1: Packets will wait for the SOF token after the TxPktRdy set time before sending packets. If the IN token is received before the SOF token, a zero-length packet is sent
[6]	RW	**Soft Conn:** Soft connect/disconnect function mode. 1: Connect DP/DM signal; 0: disconnect DP/DM signal
[5]	RW	**HS Enab:** The Reserved high-speed mode is enabled by default. The value must be set to 0 when the controller is configured
[4]	R	**HS Mode:** The high - speed mode negotiation succeeds
[3]	R	**Reset:** High indicates that the reset command is received
[2]	RW	**Resume:** Restore the enable 1: Exits the suspended mode. 0: End Restore(wait 10ms-15ms after write 1)

[1]	R	**Suspend Mode:** A high value indicates that the system is in suspended mode and can resume by writing 'Resume'
[0]	RW	Reserved

25.5.3 IntrTx

Register	Address offset	Access	Reset value
IntrTx	0x02	R	0x0000_0000

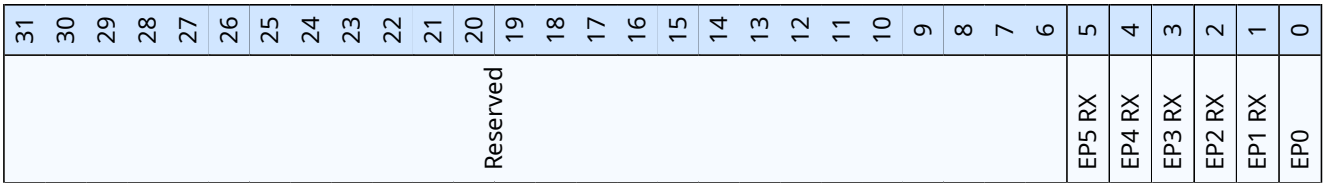


IntrTx register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	R	**EP5 TX :** Endpoint 5 TX interrupt, read after clear zero.
[4]	R	**EP4 TX :** Endpoint 4 TX interrupt, read after clear zero.
[3]	R	**EP3 TX :** Endpoint 3 TX interrupt, read after clear zero.
[2]	R	**EP2 TX :** Endpoint 2 TX interrupt, read after clear zero.
[1]	R	**EP1 TX :** Endpoint 1 TX interrupt, read after clear zero.
[0]	R	**EP0 :** Endpoint 0 TX interrupt, read after clear zero.

25.5.4 IntrRx

Register	Address offset	Access	Reset value
IntrRx	0x04	R	0x0000_0000



IntrRx register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	R	**EP5 RX :** Endpoint 5 RX interrupt, read after clear zero.
[4]	R	**EP4 RX :** Endpoint 4 RX interrupt, read after clear zero.
[3]	R	**EP3 RX :** Endpoint 3 RX interrupt, read after clear zero.
[2]	R	**EP2 RX :** Endpoint 2 RX interrupt, read after clear zero.
[1]	R	**EP1 RX :** Endpoint 1 RX interrupt, read after clear zero.
[0]	R	Reserved

25.5.5 IntrTxE

Register	Address offset	Access	Reset value
IntrTxE	0x06	R	0x0000_003F

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
Reserved																										EP5 TXE	EP4 TXE	EP3 TXE	EP2 TXE	EP1 TXE	EP0 E

IntrTxE register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	RW	**EP5 TXE :** USB interrupt endpoint 5 TX interrupt enabled, set 1 after the breakpoint 5 TX interrupt generation will be sent to the CPU via USB interrupt, this does not affect the IntrTX.
[4]	RW	**EP4 TXE :** USB interrupt endpoint 4 TX interrupt enabled, set 1 after the breakpoint 4 TX interrupt generation will be sent to the CPU via USB interrupt, this does not affect the IntrTX.
[3]	RW	**EP3 TXE :** USB interrupt endpoint 3 TX interrupt enabled, set 1 after the breakpoint 3 TX interrupt generation will be sent to the CPU via USB interrupt, this does not affect the IntrTX.
[2]	RW	**EP2 TXE :** USB interrupt endpoint 2 TX interrupt enabled, set 1 after the breakpoint 2 TX interrupt generation will be sent to the CPU via USB interrupt, this does not affect the IntrTX.
[1]	RW	**EP1 TXE :** USB interrupt endpoint 1 TX interrupt enabled, set 1 after the breakpoint 1 TX interrupt generation will be sent to the CPU via USB interrupt, this does not affect the IntrTX.
[0]	R	**EP0 E :** USB interrupt endpoint 0 TX interrupt enabled, set 1 after the breakpoint 0 TX interrupt generation will be sent to the CPU via USB interrupt, this does not affect the IntrTX.

25.5.6 IntrRxE

Register	Address offset	Access	Reset value
IntrRxE	0x08	R	0x0000_003E

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
Reserved																										EP5 RXE	EP4 RXE	EP3 RXE	EP2 RXE	EP1 RXE	Reserved

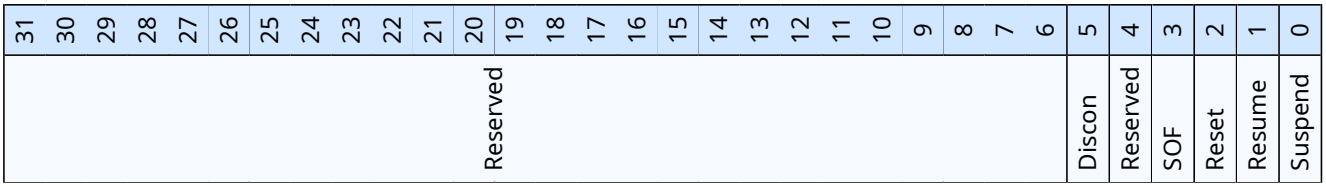
IntrRXE register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	RW	**EP5 RXE :** USB interrupt endpoint 5 RX interrupt enabled, set 1 after the breakpoint 5 RX interrupt generation will be sent to the CPU via USB interrupt, this does not affect IntrRX
[4]	RW	**EP4 RXE :** USB interrupt endpoint 4 RX interrupt enabled, set 1 after the breakpoint 4 RX interrupt generation will be sent to the CPU via USB interrupt, this does not affect IntrRX
[3]	RW	**EP3 RXE :** USB interrupt endpoint 3 RX interrupt enabled, set 1 after the breakpoint 3 RX interrupt generation will be sent to the CPU via USB interrupt, this does not affect IntrRX

[2]	RW	**EP2 RXE :** USB interrupt endpoint 2 RX interrupt enabled, set 1 after the breakpoint 2 RX interrupt generation will be sent to the CPU via USB interrupt, this does not affect IntrRX
[1]	RW	**EP1 RXE :** USB interrupt endpoint 1 RX interrupt enabled, set 1 after the breakpoint 1 RX interrupt generation will be sent to the CPU via USB interrupt, this does not affect IntrRX
[0]	R	Reserved

25.5.7 IntrUSB

Register	Address offset	Access	Reset value
IntrUSB	0x0A	R	0x0000_0000

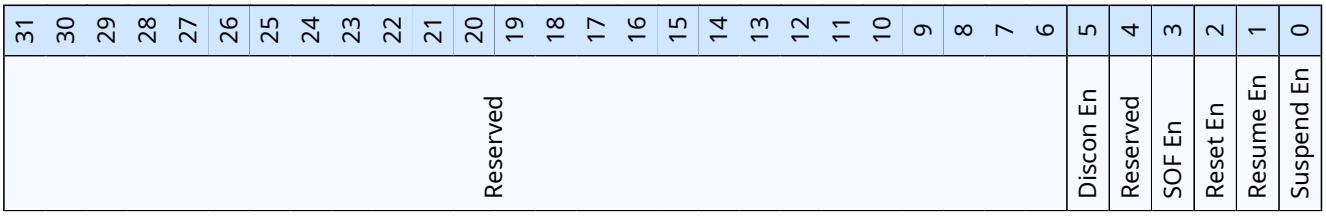


IntrUSB register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	R	**Discon:** Pull up at the end of the session and clear after reading
[4]	R	Reserved
[3]	R	**SOF:** Pull up at the start of a new frame, clear after reading
[2]	R	**Reset:** Pull high when a reset signal is detected on the bus, clear after reading
[1]	R	**EP1 RXE:** Pull high when a recovery signal is detected in hang mode and clear after reading
[0]	R	**Suspend:** Pull high when a hang signal is detected on the bus, clear after reading

25.5.8 IntrUSBE

Register	Address offset	Access	Reset value
IntrUSBE	0x0B	R	0x0000_0006



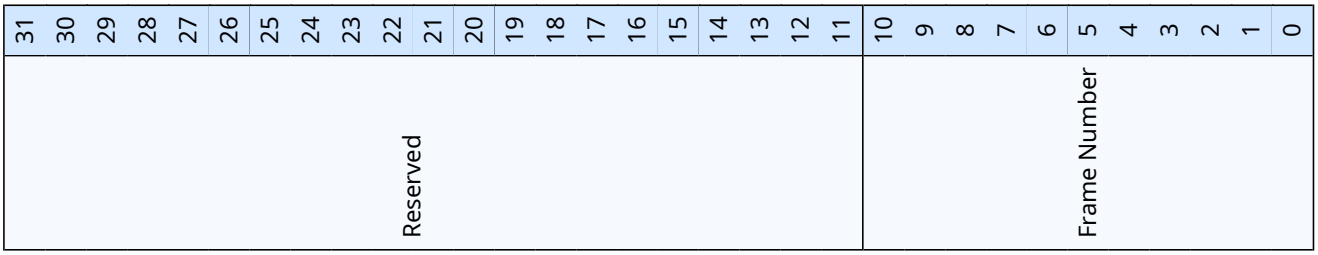
IntrUSBE register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	RW	**Discon:** USB interrupt The Discon interrupt is enabled. After setting 1, the Discon interrupt will be sent to the CPU through USB interrupt.
[4]	R	Reserved
[3]	RW	**SOF En:** USB interrupt The SOF interrupt is enabled. After 1 is set, the generated SOF interrupt is sent to the CPU through the USB interrupt

[2]	RW	**Reset En:** USB interrupt The Rese interrupt is enabled. After setting 1, the Reset interrupt is sent to the CPU through the USB interrupt.
[1]	RW	**Resume En:** USB interrupt The Resume interrupt is enabled. After setting 1, the Resume interrupt is sent to the CPU through the USB interrupt
[0]	RW	**Suspend En:** USB interrupt Enable the Suspend interrupt. After set 1, the Suspend interrupt is sent to the CPU through the USB interrupt.

25.5.9 FRAME

Register	Address offset	Access	Reset value
FRAME	0x0C	R	0x0000_0000

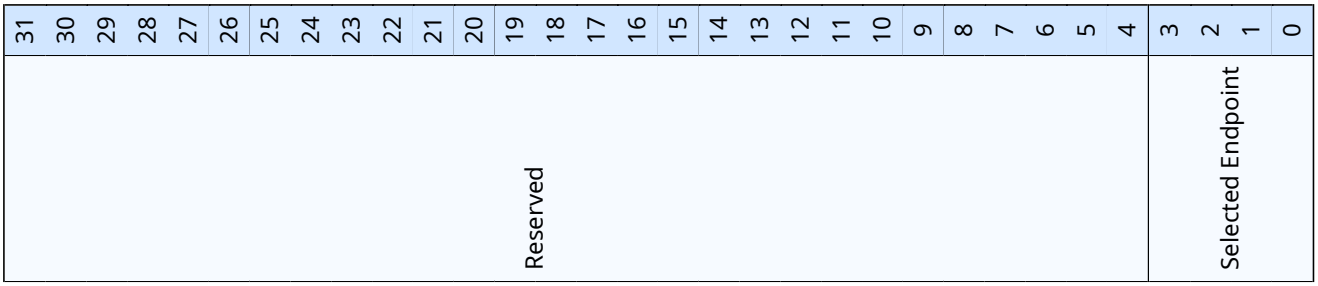


FRAME register bit description

Bit	Access	Description
[31:11]	R	Reserved
[10:0]	R	**FRAME:** The last frame number received

25.5.10 INDEX

Register	Address offset	Access	Reset value
INDEX	0x0E	RW	0x0000_0000



INDEX register bit description

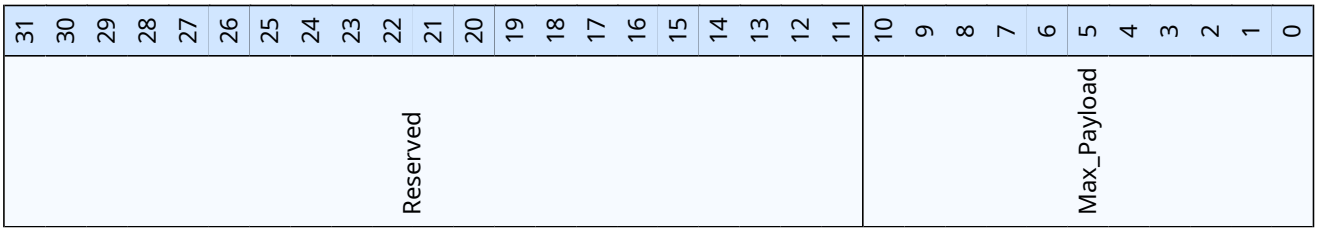
Bit	Access	Description
[31:4]	R	Reserved

[3:0]	RW	**FRAME:** Each endpoint has its own control and status register, which can be accessed separately at 0x100-0x1FF. At the same time, the control and status register of an endpoint can be mapped to the address 0x10-0x19 by selecting. Select by setting the register, for example, set to 1 after 0x10-0x19 offset address is mapped to endpoint 1 register;

25.5.11 TxMaxP (Endpoints 1-5)

Note: *TxMaxP is the maximum load length of endpoint N (N=1 to 5) TX and can be accessed at two offset addresses: 0x1N0, 0x10. Access through 0x10 requires first configuring the INDEX register to "N", which is equal to the endpoint number.*

Register	Address offset	Access	Reset value
TxMaxP	0x1N0/0x10	RW	0x0000_0000



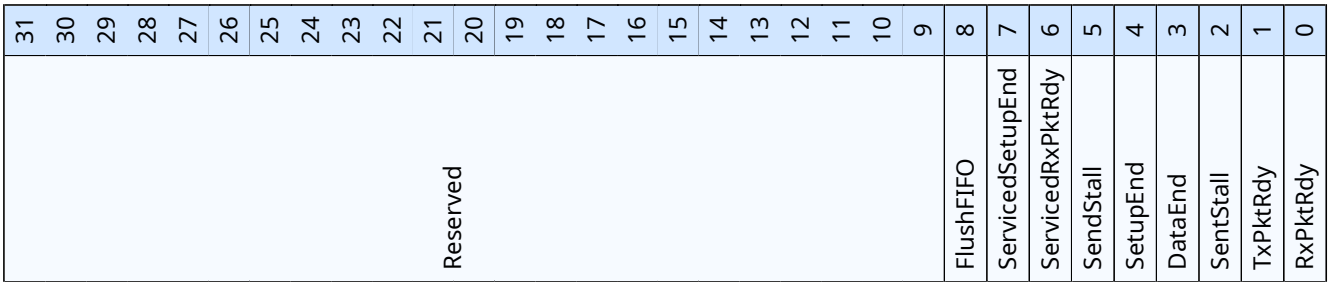
TxMaxP register bit description

Bit	Access	Description
[31:11]	R	Reserved
[10:0]	RW	**Max_Payload:** Endpoints 1-5 TX Maximum data sent at a time, in bytes

25.5.12 CSR0LH (Endpoints 0)

Note: *CSR0LH is the control/status register of endpoint 0 and can be accessed at two offset addresses: 0x102 and 0x12. Access through 0x12 requires first configuring the INDEX register to 0.*

Register	Address offset	Access	Reset value
CSR0LH	0x102/0x12	RW	0x0000_0000



CSR0LH register bit description

Bit	Access	Description
[31:9]	R	Reserved

[8]	W	**FlushFIFO:** After each write 1, the controller will generate pulse refresh FIFO, which will be performed when TxPktRdy/RxPktRdy is high, and TxPktRdy/RxPktRdy will be low after refresh.
[7]	W	**ServicedSetupEnd:** After each write 1, the controller will generate a pulse to clear the SetupEnd bit
[6]	W	**ServicedRxPktRdy:** After each write 1, the controller will generate a pulse to clear the RxPktRdy bit
[5]	W	**SendStall:** After each write 1, the controller generates a pulse to terminate the current transaction and send the STALL handshake
[4]	R	**SetupEnd:** The control transaction ends before the DataEnd has been written. The bit is pulled high, causing an interrupt and flushing the FIFO. You need to use the ServicedSetupEnd to clear this bit
[3]	R	**DataEnd:** In these cases, you need to write the bit 1 at the same time • When the last packet is sent to TxPktRdy; • Clear RxPktRdy after receiving the last packet; • Send a 0 length packet to write TxPktRdy;
[2]	RW	**SentStall:** STALL handshake after sending is complete pull up, write 0 to clear
[1]	RW	**TxPktRdy:** After sending data filling the FIFO is complete, write 1 to this bit to start sending. After sending is complete, this bit pulls down and generates an interrupt
[0]	R	**RxPktRdy:** After receiving a packet and filling the FIFO, this bit is pulled up and interrupts. At this time, the ServicedRxPktRdy can clear this bit.

25.5.13 TXCSRLH (Endpoints 1-5)

Note: *TXCSRLH is the control/status register for endpoint N (N=1 to 5) TX and can be accessed through two offset addresses: 0x1N2 and 0x12. Access through 0x12 requires first configuring the INDEX register as "N", where N equals the endpoint number.*

Register	Address offset	Access	Reset value
TXCSRLH	0x1N2/0x12	RW	0x0000_0000

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
Reserved																AutoSet	ISO	Mode	DMAReqEnab	FrcDataTog	DMAReqMode	Reserved		IncompTx	ClrDataTog	SentStall	SendStall	FlushFIFO	UnderRun	FIFONotEmpty	TxPktRdy

TXCSRLH register bit description

Bit	Access	Description
[31:16]	R	Reserved
[15]	RW	**AutoSet:** For the currently selected TX endpoint. Write 1: When the size of data stored in the FIFO exceeds TxMaxP, it automatically starts to send; otherwise, it needs to write 1 to TxPktRdy

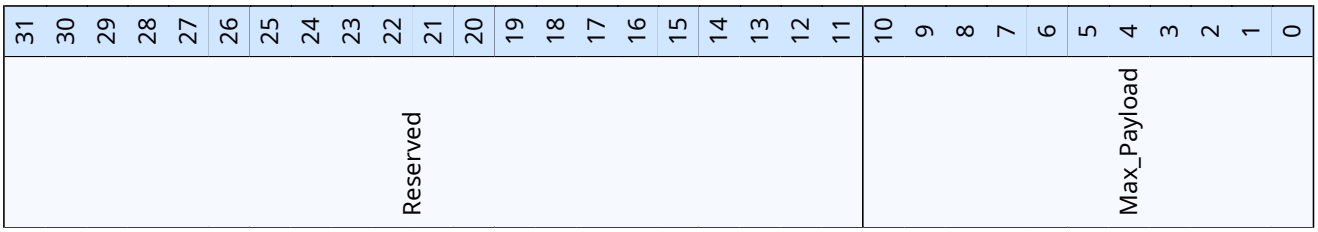
[14]	RW	**ISO:** For the currently selected TX endpoint. 1: synchronous transmission is enabled. 0: Interrupt and batch transfer are enabled.
[13]	RW	**Mode:** For the currently selected TX endpoint. 1: This endpoint is used as TX; 0: This endpoint is used as RX;
[12]	RW	**DMAReqEnab:** For the currently selected TX endpoint. 1:Open DMA channel
[11]	RW	**FrcDataTog:** For the currently selected TX endpoint. Write 1 forcibly flips data toggle and clears packets in the FIFO without waiting for ACK, which is mainly used for synchronous transmission
[10]	RW	**DMAReqMode:** For the currently selected TX endpoint. 1DMA working mode Mode 0: Interrupt is generated after a packet is transferred. This mode is used when a single packet is sent Mode 1: No interruption is generated after packets are transferred. This mode is generally used when multiple packets are sent. **Note:** *change from 1 to 0, you must first clear the DMAReqEnab to 0.*
[9:8]	RW	Reserved
[7]	RW	**IncompTx:** For the currently selected TX endpoint. During synchronous IN transmission, large data packets are split and sent IN sequence. After the first packet is sent, the bit is automatically raised. After all IN transmission is complete, 0 must be written to clear the data.
[6]	RW	**ClrDataTog:** For the currently selected TX endpoint. Write 1 to clear the data toggle bit
[5]	RW	**SentStall:** For the currently selected TX endpoint. In this case, write 1 to FlushFIFO and wait for TxPktRdy to depress the flushfifo bit before decalling the flushfifo bit
[4]	RW	**SendStall:** For the currently selected TX endpoint. Write 1 to STALL a handshake when synchronizing IN transfers. Write 0 to stall the handshake
[3]	W	**FlushFIFO:** For the currently selected TX endpoint. After each write 1, the controller will generate pulse refresh FIFO, which will be performed when TxPktRdy is high. After refresh is complete, an interrupt will be generated and TxPktRdy will be pulled down.
[2]	RW	**UnderRun:** For the currently selected TX endpoint. When the IN instruction arrives (that is, data needs to be sent), the FIFO data is not written (TxPktRdy=0).
[1]	RW	**FIFONotEmpty:** For the currently selected TX endpoint. Pull high when data is written to the FIFO
[0]	R	**TxPktRdy:** For the currently selected TX endpoint. Write 1 and send the data from the FIFO, pull low and generate an interrupt when the send is finished

25.5.14 RxMaxP (Endpoints 1-5)

Note: *RxMaxP is the maximum load length of endpoint N (N=1 to 5) RX and can be accessed at two offset addresses: 0x1N4,*

0x14. Access via 0x10 requires first configuring the INDEX register to "N", which equals the endpoint number

Register	Address offset	Access	Reset value
RxMaxP	0x1N4/0x14	RW	0x0000_0000



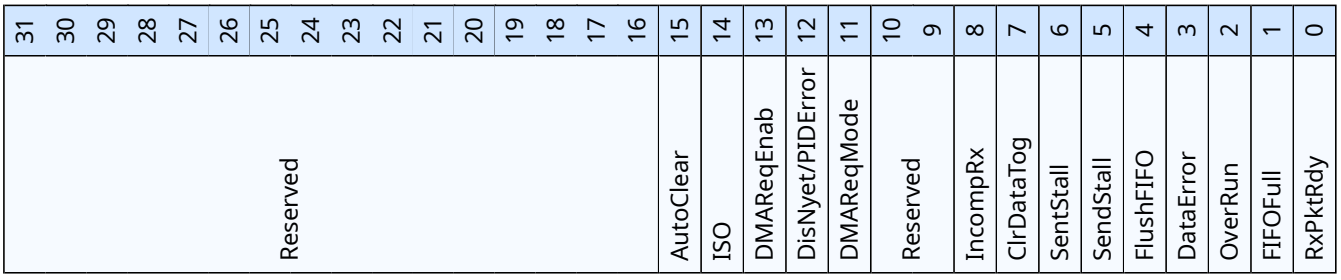
RxMaxP register bit description

Bit	Access	Description
[31:11]	R	Reserved
[10:0]	RW	**Max_Payload:** Endpoints 1-5 RX Maximum data received at a time, in bytes. **Note:** *RxMaxP must be set to an even number of bytes in order to properly generate interrupts in DMA mode 1*

25.5.15 RXCSRLH (Endpoints 1-5)

Note: *RXCSRLH is the control/status register for endpoint N (N=1 to 5) RX and can be accessed at two offset addresses: 0x1N6, 0x16. Access through 0x16 requires first configuring the INDEX register to "N", where N equals the endpoint number*

Register	Address offset	Access	Reset value
RXCSRLH	0x1N6/0x16	RW	0x0000_0000



RXCSRLH register bit description

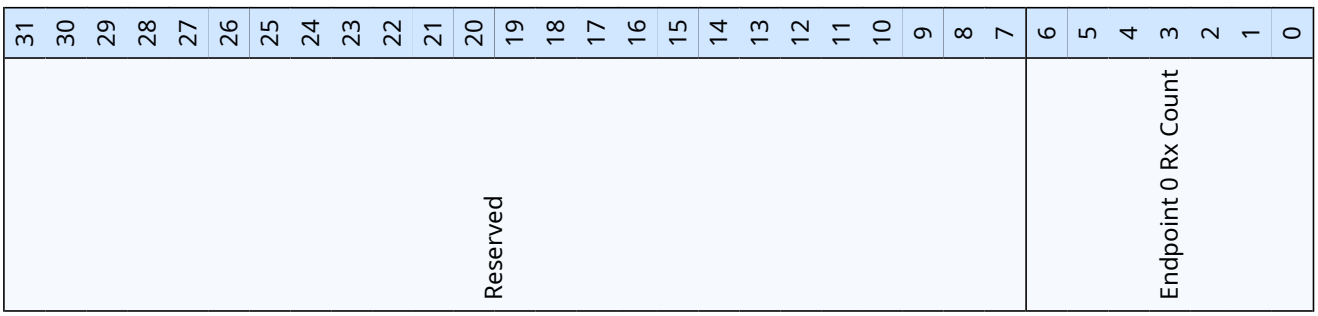
Bit	Access	Description
[31:16]	R	Reserved
[15]	RW	**AutoClear:** For the currently selected RX endpoint. After writing 1, RxPktRdy will automatically clear RxPktRdy when RXMAXP data is removed from the FIFO; otherwise, it needs to manually clear Rxpktrdy. If DMA is turned on, DMA is moved 4 bytes at a time, so the actual number of reads may be larger than RXMAXP, for example, the last fetch of 1 to 3 bytes
[14]	RW	**ISO:** For the currently selected RX endpoint. 1:Support synchronous transmission; 0: Supports interrupt and batch transfer
[13]	RW	**DMAReqEnab:** For the currently selected RX endpoint. 1:Open DMA channel

[12]	RW	**DisNyet/PIDError:** For the currently selected RX endpoint. During synchronous transmission, the PID error occurs in the received data packet. Write 1 to turn off the NYET handshake when interrupting/bulk transfers
[11]	RW	**DMAReqMode:** For the currently selected RX endpoint. DMA request mode 1, the FIFO data is larger than RxMaxP begins to move; DMA request mode 0,Start moving data when data in FIFO
[10:9]	R	Reserved
[8]	RW	**IncompRx:** For the currently selected RX endpoint. If the data is not fully received during synchronous transmission, it is raised and cleared along with RxPktRdy
[7]	RW	**ClrDataTog:** For the currently selected RX endpoint. Write 1 to clear data toggle
[6]	RW	**SentStall:** For the currently selected RX endpoint. When a STALL handshake is received, this bit is pulled high and an interrupt and a write 0 is cleared.
[5]	RW	**SendStall:** For the currently selected RX endpoint. Invalid synchronous transfer, write 1 to approve STALL handshake, can write 0 to terminate.
[4]	RW	**FlushFIFO:** For the currently selected RX endpoint. After writing 1, the controller will generate a pulse to clear the entire FIFO, including Pointers and data, which will also drag down the RxPktRdy and generate an interrupt. Be sure to empty the RxPktRdy after it has been raised
[3]	RW	**DataError:** For the currently selected RX endpoint. In synchronous transmission mode, if the packet has a CRC or bit padding error, the bit is pulled high when RxPktRdy is pulled high and cleared when RxPktRdy is cleared to zero
[2]	RW	**OverRun:** For the currently selected RX endpoint. Pull high when an OUT packet is received that cannot be written to the FIFO and write 0 to clear.
[1]	RW	**FIFOFull:** For the currently selected RX endpoint. FIFO write full and pull high
[0]	RW	**RxPktRdy:** For the currently selected RX endpoint. Pull high and generate interrupt after receiving data and writing to FIFO, write 0 can Clear

25.5.16 Count0 (Endpoints 0)

Note: *CSR0LH is an RX FIFO counter at endpoint 0 and can be accessed at two offset addresses: 0x108 and 0x18. Access through 0x18 requires first configuring the INDEX register to 0.*

Register	Address offset	Access	Reset value
Count0	0x108/0x18	R	0x0000_0000



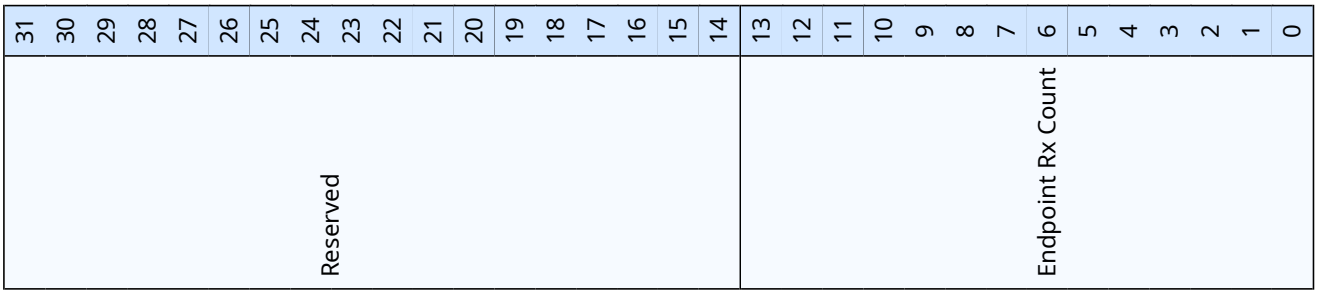
Count0 register bit description

Bit	Access	Description
[31:7]	R	Reserved
[6:0]	RW	**Endpoint 0 Rx Count:** Endpoint 0 Rx Total data received in bytes, It is valid when RxPktRdy is 1.

25.5.17 RxCount (Endpoints 1-5)

Note: *RxCount is a FIFO counter for endpoints N (N=1 to 5) RX and can be accessed at two offset addresses: 0x1N8, 0x18. Access through 0x18 requires first configuring the INDEX register to "N", which equals the endpoint number*

Register	Address offset	Access	Reset value
RxCount	0x1N8/0x18	R	0x0000_0000



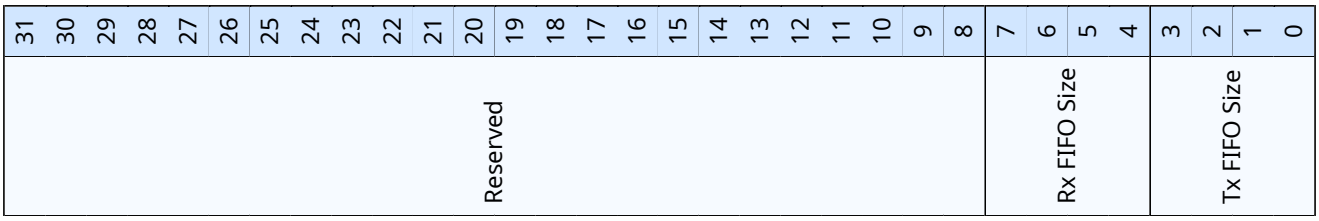
RxCount register bit description

Bit	Access	Description
[31:14]	R	Reserved
[13:0]	R	**Endpoint Rx Count:** For the currently selected RX endpoint. The total data received in FIFO is valid when RxPktRdy is 1;

25.5.18 FIFOSize (Endpoints 1-5)

Note: *FIFOSize is a FIFO size register for endpoint N (N=1 to 5). It can be accessed through two offset addresses: 0x1NF and 0x1F. Access through 0x1F requires first configuring the INDEX register as "N", where N equals the endpoint number*

Register	Address offset	Access	Reset value
FIFOSize	0x1NF/0x1F	R	0x0000_0000



RxCount register bit description

Bit	Access	Description
[31:8]	R	Reserved

[7:4]	R	**Rx FIFO Size:** FIFO Size assigned to Rx; 3:8 bytes; 4:16 bytes; .... 10:1024 bytes;
[3:0]	R	**Tx FIFO Size:** FIFO Size assigned to Tx; 3:8 bytes; 4:16 bytes; .... 10:1024 bytes;

25.5.19 FIFO0 (Endpoints 0)

Register	Address offset	Access	Reset value
FIFO0	0x20	R	0x0000_0000

FIFO0 register bit description

Bit	Access	Description
[31:0]	RW	**FIFO0:** FIFO data for endpoint 0, writable in TX and readable in RX

25.5.20 FIFO1 (Endpoints 1)

Register	Address offset	Access	Reset value
FIFO1	0x24	RW	0x0000_0000

FIFO1 register bit description

Bit	Access	Description
[31:0]	RW	**FIFO1:** FIFO data for endpoint 1, writable in TX and readable in RX

25.5.21 FIFO2 (Endpoints 2)

Register	Address offset	Access	Reset value
FIFO2	0x28	RW	0x0000_0000

FIFO2 register bit description

Bit	Access	Description
[31:0]	RW	**FIFO2:** FIFO data for endpoint 2, writable in TX and readable in RX

25.5.22 FIFO3 (Endpoints 3)

Register	Address offset	Access	Reset value
FIFO3	0x2C	RW	0x0000_0000

FIFO3 register bit description

Bit	Access	Description
[31:0]	RW	**FIFO3:** FIFO data for endpoint 3, writable in TX and readable in RX

25.5.23 FIFO4 (Endpoints 4)

Register	Address offset	Access	Reset value
FIFO4	0x30	RW	0x0000_0000

FIFO4 register bit description

Bit	Access	Description
[31:0]	RW	**FIFO4:** FIFO data for endpoint 4, writable in TX and readable in RX

25.5.24 FIFO5 (Endpoints 5)

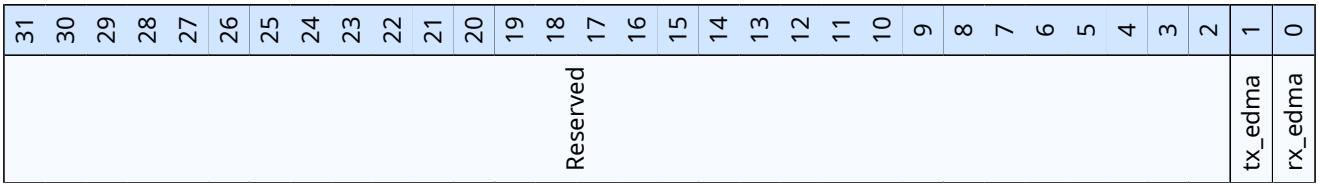
Register	Address offset	Access	Reset value
FIFO5	0x34	RW	0x0000_0000

FIFO5 register bit description

Bit	Access	Description
[31:0]	RW	**FIFO4:** FIFO data for endpoint 5, writable in TX and readable in RX

25.5.25 MISC

Register	Address offset	Access	Reset value
MISC	0x61	RW	0x0000_0000



MISC register bit description

Bit	Access	Description
[31:2]	R	Reserved
[1]	RW	**tx_edma:**All endpoints IN the IN transport release DMA requests Signal location selection; 0: MAXP release after all bytes are written. 1: MAXP-8 bytes are released after written;
[0]	RW	**rx_edma:** All endpoints OUT the IN transport release DMA requests Signal location selection; 0: MAXP release after all bytes are read. 1: MAXP-8 bytes are released after read;

25.5.26 TxFIFOsz (Endpoints 1-5)

Register	Address offset	Access	Reset value
TxFIFOsz	0x62	RW	0x0000_0000

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
Reserved																											DPB	SZ[3:0]				

TxFIFOsz register bit description

Bit	Access	Description
[31:5]	R	Reserved
[4]	RW	**DPB:**This is valid only for the selected endpoint of the current INDEX. 1: TX supports dual buffering; 0: TX supports single buffering
[3:0]	RW	**SZ[3:0]:** This is valid only for the selected endpoint of the current INDEX. The maximum packet size allowed by TX when DPB=0; 0:8 bytes; 1:16 bytes; 2:32 bytes; ... 8:2048 bytes; 9:4096 bytes;

25.5.27 RxFIFOsz (Endpoints 1-5)

Register	Address offset	Access	Reset value
RxFIFOsz	0x63	RW	0x0000_0000

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
Reserved																											DPB	SZ[3:0]				

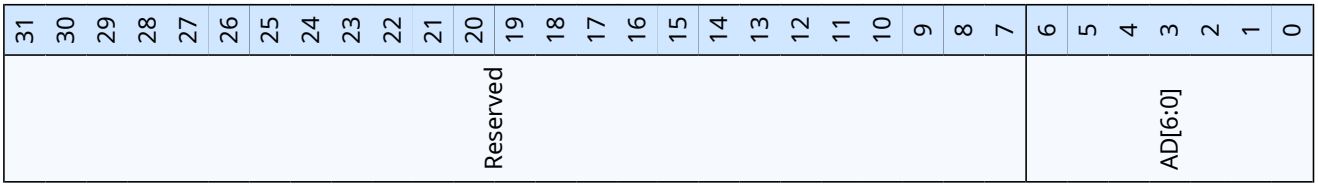
RxFIFOsz register bit description

Bit	Access	Description
[31:5]	R	Reserved
[4]	RW	**DPB:**This is valid only for the selected endpoint of the current INDEX. Only for endpoints 1-5; 1: RX supports dual buffering; 0: RX supports single buffering;

[3:0]	RW	**SZ[3:0]:**This is valid only for the selected endpoint of the current INDEX. Only for endpoints 1-5; The maximum packet size allowed by RX when DPB=0; 0:8 bytes; 1:16 bytes; 2:32 bytes; ... 8:2048 bytes; 9:4096 bytes; 10-15: Not supported When DPB=1, the value is multiplied by 2;

25.5.28 TxFIFOadd (Endpoints 0-5)

Register	Address offset	Access	Reset value
TxFIFOadd	0x64	RW	0x0000_0000

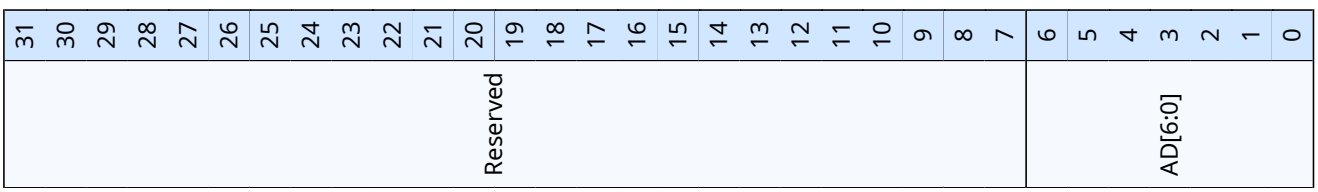


TxFIFOadd register bit description

Bit	Access	Description
[31:7]	R	Reserved
[6:0]	RW	**AD[6:0]:**This is valid only for the selected endpoint of the current INDEX. The starting address of the endpoint TX FIFO; 0:0x0000 1:0x0008 2:0x0010; ... 127:0x3F8;

25.5.29 RxFIFOadd (Endpoints 0-5)

Register	Address offset	Access	Reset value
RxFIFOadd	0x66	RW	0x0000_0000

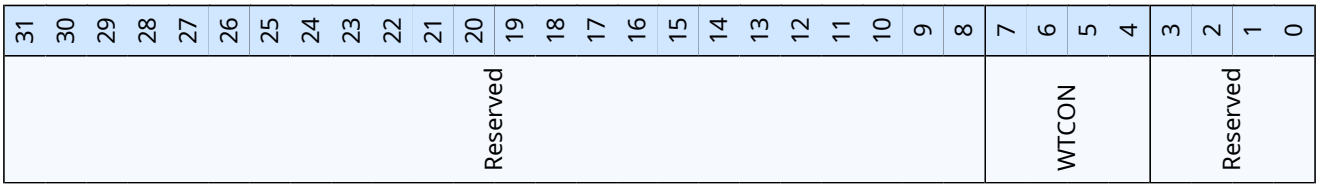


TxFIFOadd register bit description

Bit	Access	Description
[31:7]	R	Reserved
[6:0]	RW	**AD[6:0]:**This is valid only for the selected endpoint of the current INDEX. The starting address of the endpoint RX FIFO; 0:0x0000 1:0x0008 2:0x0010; ... 127:0x3F8;

25.5.30 LinkInfo

Register	Address offset	Access	Reset value
LinkInfo	0x7a	RW	0x5C

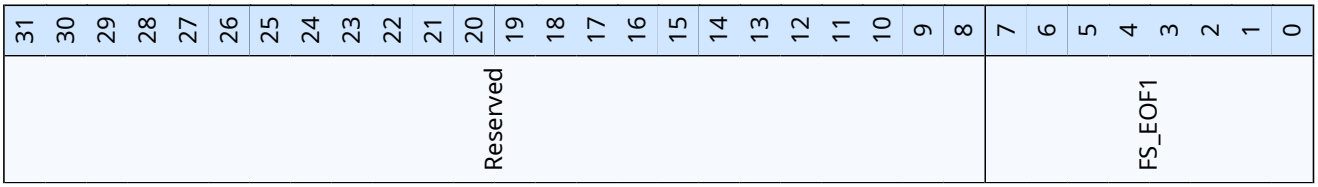


LinkInfo register bit description

Bit	Access	Description
[31:8]	R	Reserved
[7:4]	RW	**WTCON:**Connection/disconnection detection time, unit: 533.3ns. (By default, 5*533.3ns= 2.667μs.)
[3:0]	R	Reserved

25.5.31 FS_EOF1

Register	Address offset	Access	Reset value
FS_EOF1	0x7d	RW	0x77



FS_EOF1 register bit description

Bit	Access	Description
[31:8]	R	Reserved
[7:0]	RW	**FS_EOF1:** The wait time before the EOF is set so that the next transaction will not start. The unit is 533.3ns, and the default is 63.46 μs.

25.5.32 SOFT_RST

Register	Address offset	Access	Reset value
SOFT_RST	0x7f	RW	0x0000_0000

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
Reserved																														NRSTX	NRST

SOFT_RST register bit description

Bit	Access	Description
[31:2]	R	Reserved
[1]	RW	**NRSTX:** USB PHY interface software reset, reset after write 1, automatically clear 0
[0]	RW	**NRST:** USB system interface software reset, reset after write 1, automatically clear 0

25.5.33 DMA_INTR

Register	Address offset	Access	Reset value
DMA_INTR	0x200	RW	0x0000_0000

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
Reserved																												CH4_DMA_INTR	CH3_DMA_INTR	CH2_DMA_INTR	CH1_DMA_INTR

DMA_INTR register bit description

Bit	Access	Description
[31:4]	R	Reserved
[3]	RW	**CH4_DMA_INTR:** DMA channel 4 interrupt status, clear 0 when read
[2]	RW	**CH3_DMA_INTR:** DMA channel 3 interrupt status, clear 0 when read
[1]	RW	**CH2_DMA_INTR:** DMA channel 2 interrupt status, clear 0 when read
[0]	RW	**CH1_DMA_INTR:** DMA channel 1 interrupt status, clear 0 when read

25.5.34 DMA_CH1_CNTL

Register	Address offset	Access	Reset value
DMA_CH1_CNTL	0x204	RW	0x0000_0000

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
Reserved																					DMA_BRSTM		DMA_ERR	DMAEP				DMAIE	DMAMODE	DMA_DIR	DMA_ENAB

DMA_CH1_CNTL register bit description

Bit	Access	Description
[31:11]	R	Reserved
[10:9]	RW	**DMA_BRSTM:** The DMA channel accesses the bus mode; 00 = single 01 = INCR4 10 = INCR8, INCR4 11 = INCR16, INCR8, INCR4
[8]	RW	**DMA_ERR:** An error occurred on this DMA channel access bus.
[7:4]	RW	**DMAEP:** The endpoint number assigned to this DMA channel
[3]	RW	**DMAIE:** Write 1: DMA interrupt on
[2]	RW	**DMAMODE:** This DMA channel transfer mode. 0 = DMA mode 0 transfer 1 = DMA mode 1 transfer
[1]	RW	**DMA_DIR:** Direction of transmission for this DMA channel. 0 = DMA write outwards (RX endpoint) 1 = DMa read from outside (TX endpoint)
[0]	RW	**DMA_ENAB:** Write 1 to start this DMA channel

25.5.35 DMA_CH1_ADDR

Register	Address offset	Access	Reset value
DMA_CH1_ADDR	0x208	RW	0x0000_0000

DMA_CH1_ADDR register bit description

Bit	Access	Description
[31:0]	RW	**DMA_ADDR:** The DMA channel accesses the start address, with the low 2 bits fixed at 0

25.5.36 DMA_CH1_COUNT

Register	Address offset	Access	Reset value
DMA_CH1_COUNT	0x20C	RW	0x0000_0000

DMA_CH1_COUNT register bit description

Bit	Access	Description
[31:0]	RW	**DMA_COUNT:** The NTH DMA channel control. The DMA channel moves the data length and the hardware behavior decays automatically after the initial value is written

25.5.37 DMA_CH2_CNTL

Register	Address offset	Access	Reset value
DMA_CH2_CNTL	0x214	RW	0x0000_0000

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
Reserved																					DMA_BRSTM		DMA_ERR	DMAEP				DMAIE	DMAMODE	DMA_DIR	DMA_ENAB

DMA_CH2_CNTL register bit description

Bit	Access	Description
[31:11]	R	Reserved
[10:9]	RW	**DMA_BRSTM:** The DMA channel accesses the bus mode; 00 = single 01 = INCR4 10 = INCR8, INCR4 11 = INCR16, INCR8, INCR4
[8]	RW	**DMA_ERR:** An error occurred on this DMA channel access bus.
[7:4]	RW	**DMAEP:** The endpoint number assigned to this DMA channel
[3]	RW	**DMAIE:** Write 1: DMA interrupt on
[2]	RW	**DMAMODE:** This DMA channel transfer mode. 0 = DMA mode 0 transfer 1 = DMA mode 1 transfer
[1]	RW	**DMA_DIR:** Direction of transmission for this DMA channel. 0 = DMA write outwards (RX endpoint) 1 = DMa read from outside (TX endpoint)
[0]	RW	**DMA_ENAB:** Write 1 to start this DMA channel

25.5.38 DMA_CH2_ADDR

Register	Address offset	Access	Reset value
DMA_CH2_ADDR	0x218	RW	0x0000_0000

DMA_CH2_ADDR register bit description

Bit	Access	Description
[31:0]	RW	**DMA_ADDR:** The DMA channel accesses the start address, with the low 2 bits fixed at 0

25.5.39 DMA_CH2_COUNT

Register	Address offset	Access	Reset value
DMA_CH2_COUNT	0x21C	RW	0x0000_0000

DMA_CH2_COUNT register bit description

Bit	Access	Description

[31:0]	RW	**DMA_COUNT:** The NTH DMA channel control. The DMA channel moves the data length and the hardware behavior decays automatically after the initial value is written

25.5.40 DMA_CH3_CNTL

Register	Address offset	Access	Reset value
DMA_CH3_CNTL	0x224	RW	0x0000_0000

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
Reserved																					DMA_BRSTM		DMA_ERR	DMAEP				DMAIE	DMAMODE	DMA_DIR	DMA_ENAB

DMA_CH3_CNTL register bit description

Bit	Access	Description
[31:11]	R	Reserved
[10:9]	RW	**DMA_BRSTM:** The DMA channel accesses the bus mode; 00 = single 01 = INCR4 10 = INCR8, INCR4 11 = INCR16, INCR8, INCR4
[8]	RW	**DMA_ERR:** An error occurred on this DMA channel access bus.
[7:4]	RW	**DMAEP:** The endpoint number assigned to this DMA channel
[3]	RW	**DMAIE:** Write 1: DMA interrupt on
[2]	RW	**DMAMODE:** This DMA channel transfer mode. 0 = DMA mode 0 transfer 1 = DMA mode 1 transfer
[1]	RW	**DMA_DIR:** Direction of transmission for this DMA channel. 0 = DMA write outwards (RX endpoint) 1 = DMa read from outside (TX endpoint)
[0]	RW	**DMA_ENAB:** Write 1 to start this DMA channel

25.5.41 DMA_CH3_ADDR

Register	Address offset	Access	Reset value
DMA_CH3_ADDR	0x228	RW	0x0000_0000

DMA_CH3_ADDR register bit description

Bit	Access	Description
[31:0]	RW	**DMA_ADDR:** The DMA channel accesses the start address, with the low 2 bits fixed at 0

25.5.42 DMA_CH3_COUNT

Register	Address offset	Access	Reset value
DMA_CH3_COUNT	0x22C	RW	0x0000_0000

DMA_CH3_COUNT register bit description

Bit	Access	Description
[31:0]	RW	**DMA_COUNT:** The NTH DMA channel control. The DMA channel moves the data length and the hardware behavior decays automatically after the initial value is written

25.5.43 DMA_CH4_CNTL

Register	Address offset	Access	Reset value
DMA_CH4_CNTL	0x234	RW	0x0000_0000

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
Reserved																					DMA_BRSTM		DMA_ERR	DMAEP				DMAIE	DMAMODE	DMA_DIR	DMA_ENAB

DMA_CH4_CNTL register bit description

Bit	Access	Description
[31:11]	R	Reserved
[10:9]	RW	**DMA_BRSTM:** The DMA channel accesses the bus mode; 00 = single 01 = INCR4 10 = INCR8, INCR4 11 = INCR16, INCR8, INCR4
[8]	RW	**DMA_ERR:** An error occurred on this DMA channel access bus.
[7:4]	RW	**DMAEP:** The endpoint number assigned to this DMA channel
[3]	RW	**DMAIE:** Write 1: DMA interrupt on
[2]	RW	**DMAMODE:** This DMA channel transfer mode. 0 = DMA mode 0 transfer 1 = DMA mode 1 transfer
[1]	RW	**DMA_DIR:** Direction of transmission for this DMA channel. 0 = DMA write outwards (RX endpoint) 1 = DMa read from outside (TX endpoint)
[0]	RW	**DMA_ENAB:** Write 1 to start this DMA channel

25.5.44 DMA_CH4_ADDR

Register	Address offset	Access	Reset value
DMA_CH4_ADDR	0x238	RW	0x0000_0000

DMA_CH4_ADDR register bit description

Bit	Access	Description
[31:0]	RW	**DMA_ADDR:** The DMA channel accesses the start address, with the low 2 bits fixed at 0

25.5.45 DMA_CH4_COUNT

Register	Address offset	Access	Reset value
DMA_CH4_COUNT	0x23C	RW	0x0000_0000

DMA_CH4_COUNT register bit description

Bit	Access	Description
[31:0]	RW	**DMA_COUNT:** The NTH DMA channel control. The DMA channel moves the data length and the hardware behavior decays automatically after the initial value is written

25.5.46 RxDPktBufDis

Register	Address offset	Access	Reset value
RxDPktBufDis	0x340	RW	0x0000_0000

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
Reserved																										EP5 RxDis	EP4 RxDis	EP3 RxDis	EP2 RxDis	EP1 RxDis	Reserved

RxDPktBufDis register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	RW	**EP5 RxDis:** Rx double buffered switch at endpoint 5, write 1 disabled
[4]	RW	**EP4 RxDis:** Rx double buffered switch at endpoint 4, write 1 disabled
[3]	RW	**EP3 RxDis:** Rx double buffered switch at endpoint 3, write 1 disabled
[2]	RW	**EP2 RxDis:** Rx double buffered switch at endpoint 2, write 1 disabled
[1]	RW	**EP1 RxDis:** Rx double buffered switch at endpoint 1, write 1 disabled
[0]	RW	Reserved

25.5.47 TxDPktBufDis

Register	Address offset	Access	Reset value
TxDPktBufDis	0x342	RW	0x0000_0000

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
Reserved																										EP5 TxDis	EP4 TxDis	EP3 TxDis	EP2 TxDis	EP1 TxDis	Reserved

TxDPktBufDis register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	RW	**EP5 TxDis:** Tx double buffered switch at endpoint 5, write 1 disabled
[4]	RW	**EP4 TxDis:** Tx double buffered switch at endpoint 4, write 1 disabled
[3]	RW	**EP3 TxDis:** Tx double buffered switch at endpoint 3, write 1 disabled
[2]	RW	**EP2 TxDis:** Tx double buffered switch at endpoint 2, write 1 disabled
[1]	RW	**EP1 TxDis:** Tx double buffered switch at endpoint 1, write 1 disabled
[0]	RW	Reserved

25.5.48 LPM_ATTR

Register	Address offset	Access	Reset value
LPM_ATTR	0x360	RW	0x0000_0000

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
Reserved																EndPnt				Reserved			RmtWak	HIRD				LinkState			

LPM_ATTR register bit description

Bit	Access	Description
[31:16]	R	Reserved
[15:12]	RW	**EndPnt:** Updated after successfully receiving an LPM transaction. The endpoint of the LPM transaction command packet
[11:9]	RW	Reserved
[8]	RW	**RmtWak:** Updated after successfully receiving an LPM transaction. The remote wake up function was enabled. 0: disables remote wake up. 1: Enable remote wake up.
[7:4]	RW	**HIRD:** Updated after successfully receiving an LPM transaction. Time required for host initialization.That is, the time it takes the host to return to the bus: 50us + HIRD * 75us (50us~1200us)
[3:0]	RW	**LinkState:** Updated after successfully receiving an LPM transaction. State arranged by the host: 0/2/3: reserved; 1: Sleep state (L1)

25.5.49 LPM_CNRL

Register	Address offset	Access	Reset value
LPM_CNRL	0x362	RW	0x0000_0000

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
Reserved																											LPMNAK	LPMEN			LPMRES	LPMXMTW

LPM_CNRL register bit description

Bit	Access	Description
[31:5]	R	Reserved
[4]	RW	**LPMNAK:** Writing 1 puts all endpoints in a state where the response to all transactions except LPM transactions is NAK until the bit is cleared to zero
[3:2]	RW	**LPMEN:** LPM enable options: 2 'b00, 2' b10: No support for LPM and extended transactions 2 'b01: does not support LPM but supports extended transactions. In this case, the STALL will respond to the LPM transaction. 2 'b11: Support for LPM extension transactions. In this case, the response is NYET or ACK, depending on the value of LPMXMT and other conditions
[1]	RW	**LPMRES:** Start recovery (remote wake up). Write 1 to start recovery, 50us after the automatic clearing.
[0]	RW	**LPMXMT:** Write 1: Transition to L1 state when the next LPM transaction is received. This bit is only valid if LPMEN is set to 2'b11. This bit can be set in the same period as LPMEN, after which the response modes include: • If no data is pending (all TX FIFOs are empty), respond with an ACK. In this case, the bit will self-clear and produce a software interrupt. • If the data is pending (the data resides in at least one TX FIFO), an NYET is responded to. In this case, the bit does not self-clear, but a software interrupt is generated.

25.5.50 LPM_INTERN

Register	Address offset	Access	Reset value
LPM_INTERN	0x363	RW	0x0000_0000

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
Reserved																										LPMERREN	LPMRESEN	LPMNCEN	LPMACKEN	LPMNYEN	LPMSTEN

LPM_INTERN register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	RW	**LPMERREN:** 0: disables LPMERR interruption 1: LPMERR interrupt is enabled

[4]	RW	**LPMRESEN:** 0: disables LPMERR interruption 1: LPMERR interrupt is enabled
[3]	RW	**LPMNCEN:** 0: disables LPMERR interruption 1: LPMERR interrupt is enabled
[2]	RW	**LPMACKEN:** 0: disables LPMERR interruption 1: LPMERR interrupt is enabled
[1]	RW	**LPMNYEN:** 0: disables LPMERR interruption 1: LPMERR interrupt is enabled
[0]	RW	**LPMSTEN:** 0: disables LPMERR interruption 1: LPMERR interrupt is enabled

25.5.51 LPM_INTR

Register	Address offset	Access	Reset value
LPM_INTR	0x363	RW	0x0000_0000

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
Reserved																										LkPMERR	LPMRES	LPMNC	LPMACK	LPMNY	LPMST

LPM_INTR register bit description

Bit	Access	Description
[31:6]	R	Reserved
[5]	R	**LkPMERR:** LPM transaction received contains unsupported LinkState field pulled high. In this case, the response to the transaction would be the STALL. However, the LPM_ATTR register will be updated so that the software can observe nonconforming LPM packet payloads
[4]	R	**LPMRES:** When pulled high after recovery for any reason, this bit is mutually exclusive with the RESUME bit in the power repository (address 0x01)
[3]	R	**LPMNC:** Pull up when an LPM transaction is received and an NYET response is received due to unfetched data in RXFIFO.
[2]	R	**LPMACK:** Pull high when an LPM transaction is received and responds with an ACK
[1]	R	**LPMNY:** Pull high when an LPM transaction is received and responds with an NYET
[0]	R	**LPMST:** Pull high when an LPM transaction is received and responds with an STALL

26 True Random Number Generator(TRNG)

26.1 TRNG Function description

TRNG is a true random number generator that continuously provides 32-bit entropy samples.

TRNG provides 32-bit true random numbers that are generated by an simulated entropy source post-processed by a linear feedback shift register (LFSR).

One 32-bit random sample is generated every 42 RNG clock cycles (dedicated clocks).

It allows continuous condition monitoring and associated error management. Including low sampling clock detection and repeat count test.

Can be disabled to reduce power consumption.

It has the AMBA AHB interface and can only be accessed with a single 32-bit word access (otherwise an AHB bus error is generated). Any write operation that is not equal to 32 bits may corrupt register contents

26.2 TRNG Register

Tab 26.2-1 TRNG register map

offset	Register	Reset value	Description
TRNG_BA = 0x4002_5000			
0x00	RNG_CR	0x0000_0000	TRNG Control Register
0x04	RNG_SR	0x0000_0000	TRNG Status register
0x08	RNG_DR	0x0000_0000	TRNG Data register

26.2.1 TRNG Control Register(RNG_CR)

Register	Address offset	Access	Reset value	Description
RNG_CR	0x00	RW	0x0000_0000	TRNG Control Register

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

TRNG control register(RNG_CR) bit description

Bit	Access	Description
[31：6]	R	Reserved
[5]	RW	**CED:** Clock error detection 0: Clock error detection is enabled 1: Clock error detection is disabled When RNG is enabled, clock error detection cannot be enabled or disabled immediately. That is, to enable or disable CED, RNG must be disabled
[4]	R	Reserved
[3]	RW	**IE:** Interrupt enablement 0: disables RNG interrupt 1: RNG interrupt enabled. As long as DRDY = ’1’, SEIS = ’1’, or CEIS = ’1’ in the RNG_SR register, the interrupt will hang
[2]	RW	**RNGEN:** True random number generator enabled 0: true random number generator is turned off, analog noise source is turned off, and RNG clock is logically gated. 1: Enable true random number generator.
[1:0]	R	Reserved

26.2.2 TRNG Status register(RNG_SR)

Register	Address offset	Access	Reset value	Description
RNG_SR	0x04	RW	0x0000_0000	TRNG Status register

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

TRNG Status register(RNG_SR) bit description

Bit	Access	Description

[31:7]	R	Reserved
[6]	W	**SEIS:** Seed error interrupt state This bit is set at the same time as SECS. Clear it by writing it to 0. 0: No error sequence was detected 1: At least one error sequence is detected, as described in the SECS bit description. If IE = '1' in the RNG_CR register, the interrupt is suspended.
[5]	W	**CEIS:** Clock error Interrupt status This bit is set at the same time as CECS. Clear it by writing it to 0. 0: RNG clock correct (fRNGCLK > fHCLK / 16) 1: RNG detected too slow (fRNGCLK < fHCLK / 16) If IE = '1' in the RNG_CR register, the interrupt is hanging.
[4:3]	R	Reserved
[2]	R	**SECS:** Seed error Current status 0: No fault sequence is currently detected If the SEIS bit is set, it means that the fault sequence was detected and the condition has recovered. 1: One of the noise sources provides more than 64 consecutive bits at a constant value (" 0 "or" 1 "), or more than 32 consecutive two-bit patterns (" 01 "or" 10 ")
[1]	R	**CECS:** Clock error Current status 0: The RNG clock is correct (fRNGCLK> fHCLK / 16). If the CEIS bit is set, it means that a slow clock has been detected and the condition has recovered. 1: RNG clock is too slow (fRNGCLK <fHCLK / 16) **Note:** *The CECS bit is only valid if the CED bit in the RNG_CR register is set to "0".*
[0]	R	**DRDY:** Data ready 0: The RNG_DR register has not taken effect, and no random data is available. 1: The RNG_DR register contains valid random data. **Note:** *Once the RNG_DR register is read, this bit returns 0 until a new random value is generated. If IE = 1 in the RNG_CR register, an interrupt is generated when DRDY = 1*

26.2.3 TRNG Data register(RNG_DR)

Register	Address offset	Access	Reset value	Description
RNG_DR	0x08	R	0x0000_0000	TRNG data register

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

TRNG data register(RNG_DR) bit description

Bit	Access	Description
[31:0]	R	**RNDATA:** Random data: 32-bit random data when DRDY= '1' The RNDATA value is 0 when DRDY= '0'.

27 Quad-SPI interface (QSPI)

27.1 QSPI Introduction

The QUADSPI is a specialized communication interface targeting single, dual- or quad-SPI flash memories. It can operate in any of the three following modes:

- indirect mode: all the operations are performed using the QUADSPI registers.
- automatic status-polling mode: the external flash memory status register is periodically read and an interrupt can be generated in case of flag setting.
- memory-mapped mode: the external flash memory is mapped to the device address space and is seen by the system as if it was an internal memory.

Both throughput and capacity can be increased two-fold using dual-flash mode, where two Quad-SPI flash memories are accessed simultaneously.

27.2 QUADSPI main features

- Three functional modes: indirect, automatic status-polling, and memory-mapped
- Dual-flash mode, where 8 bits can be sent/received simultaneously by accessing two flash memories in parallel
- SDR and DDR support
- Fully programmable opcode for both indirect and memory-mapped modes
- Fully programmable frame format for both indirect and memory-mapped modes
- Integrated FIFO for reception and transmission
- 8-, 16-, and 32-bit data accesses allowed
- MDMA trigger generation for FIFO threshold and transfer complete
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error

27.3 QUADSPI functional description

27.3.1 QUADSPI block diagram

Fig 27.3-1 QUADSPI block diagram when dual-flash mode is disabled

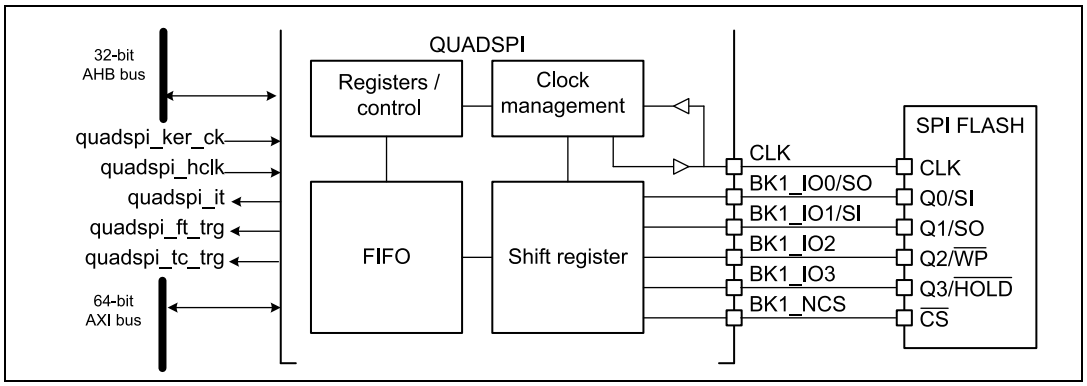
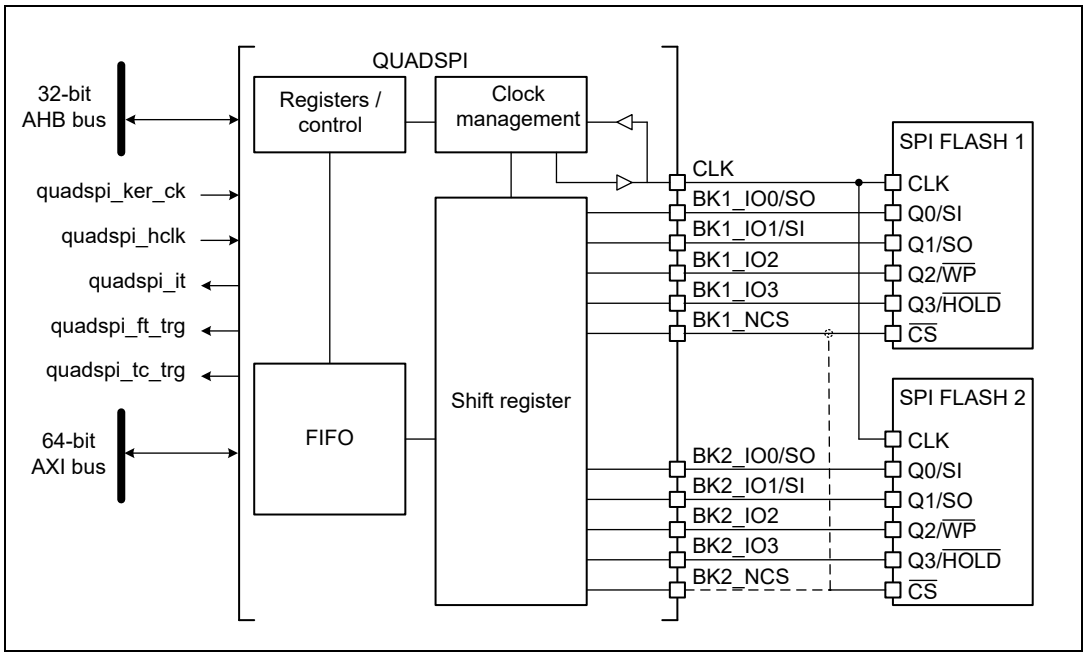


Fig 27.3-2 QUADSPI block diagram when dual-flash mode is enabled



27.3.2 QUADSPI pins and internal signals

Tab 27.3-1 QUADSPI internal signals

Signal name	Signal type	Description
quadspi_ker_ck	Digital input	QUADSPI kernel clock
quadspi_hclk	Digital input	QUADSPI register interface clock
quadspi_it	Digital output	QUADSPI global interrupt
quadspi_ft_trg	Digital output	QUADSPI FIFO threshold trigger for MDMA
quadspi_tc_trg	Digital output	QUADSPI transfer complete trigger for MDMA

The table below lists the QUADSPI pins, six for interfacing with a single flash memory, or 10 to 11 for interfacing with two flash memories (FLASH 1 and FLASH 2) in dual-flash mode.

Tab 27.3-2 QUADSPI pins

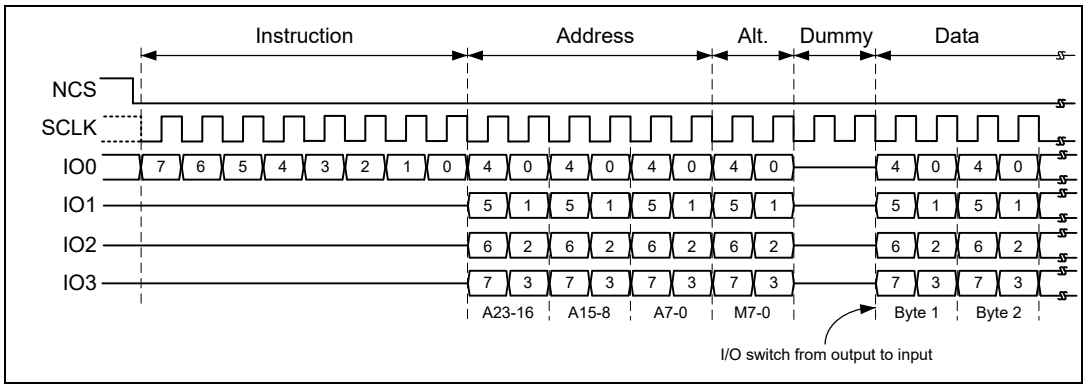
Signal name	Signal type	Description
CLK	Digital output	Clock to FLASH 1 and FLASH 2
BK1_IO0/SO	Digital input/output	Bidirectional I/O in dual/quad modes or serial output in single mode, for FLASH 1
BK1_IO1/SI	Digital input/output	Bidirectional I/O in dual/quad modes or serial input in single mode, for FLASH 1
BK1_IO2	Digital input/output	Bidirectional I/O in quad mode, for FLASH 1
BK1_IO3	Digital input/output	Bidirectional I/O in quad mode, for FLASH 1
BK2_IO0/SO	Digital input/output	Bidirectional I/O in dual/quad modes or serial output in single mode, for FLASH 2
BK2_IO1/SI	Digital input/output	Bidirectional I/O in dual/quad modes or serial input in single mode, for FLASH 2
BK2_IO2	Digital input/output	Bidirectional I/O in quad mode, for FLASH 2
BK2_IO3	Digital input/output	Bidirectional I/O in quad mode, for FLASH 2
BK1_NCS	Digital output	Chip select (active low) for FLASH 1. Can also be used for FLASH 2 if QUADSPI is always used in dual-flash mode.
BK2_NCS	Digital output	Chip select (active low) for FLASH 2. Can also be used for FLASH 1 if QUADSPI is always used in dual-flash mode.

27.3.3 QUADSPI command sequence

The QUADSPI communicates with the flash memory using commands. Each command can include five phases: instruction, address, alternate byte, dummy, data. Any of these phases can be configured to be skipped, but at least one of the instruction, address, alternate byte, or data phase must be present.

NCS falls before the start of each command and rises again after each command finishes.

Fig 27.3-3 Example of read command in quad-SPI mode



Instruction phase

During this phase, an 8-bit instruction, configured in INSTRUCTION bitfield of QUADSPI_CCR[7:0] register, is sent to the flash memory, specifying the type of operation to be performed.

Most flash memories can receive instructions only one bit at a time from the IO0/SO signal (single-SPI mode), the instruction phase can optionally send 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the IMODE[1:0] bitfield of QUADSPI_CCR[9:8] register.

When IMODE = 00, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

Address phase

In the address phase, 1-4 bytes are sent to the flash memory to indicate the address of the operation. The number of address bytes to be sent is configured in the ADSIZE[1:0] bitfield of QUADSPI_CCR[13:12] register. In indirect and automatic status-polling modes, address bytes to be sent are specified in the ADDRESS[31:0] bitfield of QUADSPI_AR register, while in memory-mapped mode, the address is given directly via the AXI (from the Cortex or from a DMA).

The address phase can send 1 bit at a time (over SO in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ADMODE[1:0] bitfield of QUADSPI_CCR[11:10] register.

When ADMODE = 00, the address phase is skipped, and the command sequence proceeds directly to the next phase, if any.

Alternate-byte phase

In the alternate-byte phase, 1-4 bytes are sent to the flash memory, generally to control the mode of operation. The number of alternate bytes to be sent is configured in the [1:0] bitfield of QUADSPI_CCR[17:16] register. The bytes to be sent are specified in the QUADSPI_ABR register.

The alternate-bytes phase can send 1 bit at a time (over SO in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ABMODE[1:0] bitfield of QUADSPI_CCR[15:14] register.

When ABMODE = 00, the alternate-byte phase is skipped, and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-mode is used and only two cycles are used for the alternate bytes. In this case, the firmware can use quad-mode (ABMODE = 11) and send a byte with bits 7 and 3 of ALTERNATE set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low). In this case, the upper two bits of the nibble to be sent are placed in bits 4:3 of ALTERNATE, while the lower two bits are placed in bits 1 and 0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE must be set to 0x8A (1000_1010).

Dummy-cycle phase

In the dummy-cycle phase, 1-31 cycles are given without any data being sent or received, in order to give time to the flash memory to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified in the DCYC[4:0] bitfield of QUADSPI_CCR[22:18] register. In both SDR and DDR modes, the duration is specified as a number of full CLK cycles.

When DCYC is zero, the dummy-cycles phase is skipped, and the command sequence proceeds directly to the data phase, if present.

The operating mode of the dummy-cycles phase is determined by DMODE.

In order to assure enough "turn-around" time for changing data signals from output mode to input mode, there must be at least one dummy cycle when using dual or quad mode to receive data from the flash memory.

Data phase

During the data phase, any number of bytes can be sent to, or received from the flash memory.

In indirect and automatic status-polling modes, the number of bytes to be sent/received is specified in the QUADSPI_DLR register.

In indirect-write mode the data to be sent to the flash memory must be written to the QUADSPI_DR register. In indirect-read mode the data received from the flash memory is obtained by reading the QUADSPI_DR register.

In memory-mapped mode, the data which is read is sent back directly over the AXI to the Cortex or to a DMA.

The data phase can send/receive 1 bit at a time (over SO/SI in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ABMODE[1:0] bitfield of QUADSPI_CCR[15:14] register.

When DMODE = 00, the data phase is skipped, and the command sequence finishes immediately by raising NCS. This configuration must only be used in only indirect write mode.

27.3.4 QUADSPI signal interface protocol modes

Single-SPI mode

This legacy SPI mode allows just one single bit to be sent/received serially. In this mode, data are sent to the flash memory over the SO signal (whose I/O shared with IO0). Data received from the flash memory arrive via SI (whose I/O shared with IO1).

The different phases can each be configured separately to use this mode by setting the IMODE/ADMODE/ABMOD fields (in QUADSPI_CCR) to 01.

In each phase which is configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).

This is the case even for the dummy phase if DMODE = 01.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting the IMODE/ADMODE/ABM fields of QUADSPI_CCR register to 10.

In each phase which is configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.

In the dummy phase when DMODE = 01, IO0/IO1 are always high-impedance.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use quad-SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI_CCR register to 11.

In each phase which is configured in this mode, IO0/IO1/IO2/IO3 are all are at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE = 11, IO0/IO1/IO2/IO3 are all high-impedance.

IO2 and IO3 are used only in quad-SPI mode. If none of the phases are configured to use quad-SPI mode, then the pins corresponding to IO2 and IO3 can be used for other functions even while the QUADSPI is active.

SDR mode

By default, the DDRM bit (QUADSPI_CCR[31]) is 0 and the QUADSPI operates in single-data rate (SDR) mode.

In SDR mode, when the QUADSPI drives IO0/SO, IO1, IO2, IO3 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the QUADSPI assumes that flash memories also send the data using CLK falling edge. By default (when SSHIFT = 0), the signals are sampled using the following (rising) edge of CLK.

DDR mode

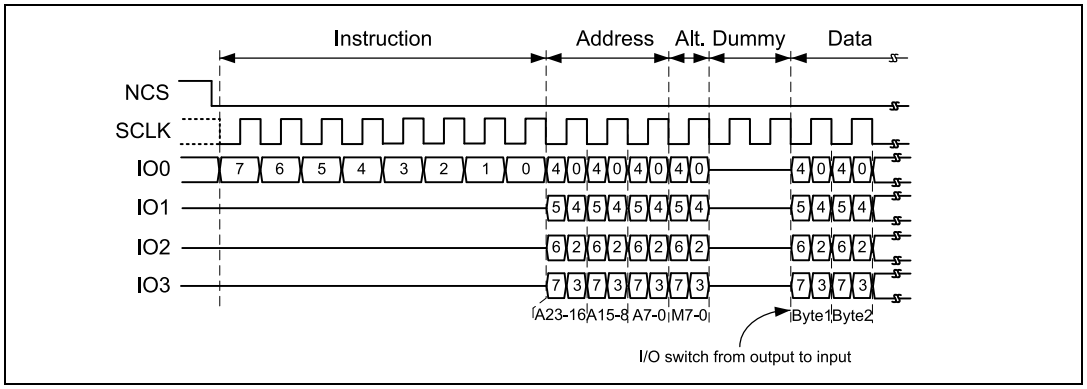
When the DDRM bit (QUADSPI_CCR[31]) is set to 1, the QUADSPI operates in double-data rate (DDR) mode.

In DDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals in the address/alternate-byte/data phases, a bit is sent on each of CLK falling and rising edges.

The instruction phase is not affected by DDRM. The instruction is always sent using CLK falling edge.

When receiving data in DDR mode, the QUADSPI assumes that flash memories also send the data using both CLK rising and falling edges. When DDRM = 1, the firmware must clear SSHIFT (bit 4 of QUADSPI_CR). Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

Fig 27.3-4 Example of a DDR command in quad-SPI mode



Dual-flash mode

When the DFM bit (bit 6 of QUADSPI_CR) is 1, the QUADSPI is in dual-flash mode, where two external quad-SPI flash memories (FLASH 1 and FLASH 2) are used in order to send/receive 8 bits (or 16 bits in DDR mode) every cycle, effectively doubling the throughput as well as the capacity.

Each of the flash memories uses the same CLK and optionally the same NCS signals, but each have separate IO0, IO1, IO2, and IO3 signals.

The dual-flash mode can be used in conjunction with single-, dual-, and quad-SPI modes, as well as with either SDR or DDR mode.

The flash memory size, as specified in FSIZE[4:0] (QUADSPI_DCR[20:16]), must reflect the total flash memory capacity, which is double the size of one individual component.

If address X is even, then the byte which the QUADSPI gives for address X is the byte at the address X/2 of FLASH 1, and the byte which the QUADSPI gives for address X+1 is the byte at the address X/2 of FLASH 2. In other words, bytes at even addresses are all stored in FLASH 1 and bytes at odd addresses are all stored in FLASH 2.

When reading the flash memories status registers in dual-flash mode, twice as many bytes must be read compared to doing the same read in single-flash mode. This means that if each flash memory

gives 8 valid bits after the instruction for fetching the status register, then the QUADSPI must be configured with a data length of 2 bytes (16 bits), and the QUADSPI receives one byte from each flash memory. If each flash memory gives a status of 16 bits, then the QUADSPI must be configured to read 4 bytes to get all the status bits of both flash memories in dual-flash mode. The least-significant byte of the result (in the data register) is the least-significant byte of FLASH 1 status register, while the next byte is the least- significant byte of FLASH 2 status register. Then, the third byte of the data register is FLASH 1 second byte, while the forth byte is FLASH 2 second byte (in the case that the flash memories have 16-bit status registers).

An even number of bytes must always be accessed in dual-flash mode. For this reason, bit 0 of the data length bitfield (QUADSPI_DLR[0]) is stuck at 1 when DFM = 1.

In dual-flash mode, the behavior of FLASH 1 interface signals are basically the same as in normal mode. FLASH 2 interface signals have exactly the same waveforms as FLASH 1 during the instruction, address, alternate-byte, and dummy-cycles phases. In other words, each flash memory always receives the same instruction and the same address. Then, during the data phase, the BK1_IOx and BK2_IOx buses are both transferring data in parallel, but the data that are sent to (or received from) FLASH 1 are distinct from those of FLASH 2.

QUADSPI indirect mode

When in indirect mode, commands are started by writing to QUADSPI registers, and data are transferred by writing or reading the data register, in the same way as for other communication peripherals.

When FMODE = 00 (QUADSPI_CCR[27:26]), the QUADSPI is in indirect-write mode, where bytes are sent to the flash memory during the data phase. Data are provided by writing to the data register (QUADSPI_DR).

When FMODE = 01, the QUADSPI is in indirect-read mode, where bytes are received from the flash memory during the data phase. Data are recovered by reading QUADSPI_DR.

The number of bytes to be read/written is specified in the data length register (QUADSPI_DLR). If QUADSPI_DLR = 0xFFFF_FFFF (all 1s), then the data length is considered undefined and the QUADSPI simply continues to transfer data until the end of flash memory (as defined by FSIZE) is reached. If no bytes are to be transferred, DMODE (QUADSPI_CCR[25:24]) must be set to 00.

If QUADSPI_DLR = 0xFFFF_FFFF and FSIZE = 0x1F (max value indicating a 4-Gbyte flash memory), then in this special case, transfers continue indefinitely, stopping only after an abort request or after the QUADSPI is disabled. After the last memory address is read (at address 0xFFFF_FFFF), reading continues with address = 0x0000_0000.

When the programmed number of bytes to be transmitted or received is reached, TCF is set and an interrupt is generated if TCIE = 1. In the case of undefined number of data, the TCF is set when the limit of the external SPI memory is reached according to the flash memory size defined in QUADSPI_CR.

Triggering the start of a command

Essentially, a command starts as soon as firmware gives the last information that is necessary for this command. Depending on the QUADSPI configuration, there are three different ways to trigger the start of a command in indirect mode. The commands starts immediately:

- after a write is performed to INSTRUCTION[7:0] (QUADSPI_CCR), if no address is necessary (when ADMODE = 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
- after a write is performed to ADDRESS[31:0] (QUADSPI_AR), if an address is necessary (when ADMODE ≠ 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
- after a write is performed to DATA[31:0] (QUADSPI_DR), if data needs to be provided by the firmware (when FMODE = 00 and DMODE != 00)

Writes to the alternate byte register (QUADSPI_ABR) never trigger the communication start. If alternate bytes are required, they must be programmed before.

As soon as a command is started, BUSY (bit 5 of QUADSPI_SR) is automatically set.

FIFO and data management

In indirect mode, data go through a 32-byte FIFO which is internal to the QUADSPI. FLEVEL[5:0] (QUADSPI_SR[13:8]) indicates how many bytes are currently being held in the FIFO.

In indirect-write mode (FMODE = 00), the firmware adds data to the FIFO when it writes QUADSPI_DR. Word writes add 4 bytes to the FIFO, halfword writes add 2 bytes, and byte writes add only 1 byte. If the firmware adds too many bytes to the FIFO (more than is indicated by DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

Byte/halfword accesses to QUADSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES[4:0] is used to define a FIFO threshold. When the threshold is reached, FTF (FIFO threshold flag) is set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there are data in the FIFO after the last byte is read from the flash memory, regardless of the FTHRES setting.

In indirect-write mode, FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when FTF is set. FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data is transferred by the CPU or DMA).

In indirect-read mode when the FIFO becomes full, the QUADSPI temporarily stops reading bytes from the flash memory to avoid an overrun. The reading of the flash memory does not restart until 4 bytes become vacant in the FIFO (when FLEVEL $\leqslant$ 11). Thus, when FTHRES $\geqslant$ 13, the application must take care to read enough bytes to assure that the QUADSPI starts retrieving data from the flash memory again. Otherwise, the FTF flag stays at 0 as long as 11 < FLEVEL < FTHRES.

27.3.5 QUADSPI automatic status-polling mode

In automatic status-polling mode, the QUADSPI periodically starts a command to read a defined number of status bytes (up to 4). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value.

Accesses to the flash memory begin in the same way as in indirect-read mode: if no address is required (AMODE = 00), accesses begin as soon as the QUADSPI_CCR is written. Otherwise, if an address is required, the first access begins when QUADSPI_AR is written. BUSY goes high at this point and stays high even between the periodic accesses.

The contents of MASK[31:0] (QUADSPI_PSMAR) are used to mask the data from the flash memory in automatic status-polling mode. If MASK[n] = 0, then bit n of the result is masked and not considered. If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] (QUADSPI_PSMAR), then there is a match for bit n.

If the polling match mode bit (PMM, bit 23 of QUADSPI_CR) is 0, then "AND" match mode is activated. This means status match flag (SMF) is set only when there is a match on all of the unmasked bits.

If PMM = 1, then "OR" match mode is activated. This means SMF is set if there is a match on any of the unmasked bits.
An interrupt is called when SMF is set if SMIE = 1.

If the automatic status-polling mode stop (APMS) bit is set, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1, and the periodic accesses continue until there is an abort or the QUADSPI is disabled (EN = 0).

The data register (QUADSPI_DR) contains the latest received status bytes (the FIFO is deactivated). The content of the data register is not affected by the masking used in the matching logic. The FTF status bit is set as soon as a new reading of the status is complete, and FTF is cleared as soon as the data is read.

27.3.6 QUADSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

It is forbidden to access the quad-SPI flash bank area before having properly configured and enabled the QUADSPI peripheral.

No more than 256 Mbytes can addressed even if the flash memory capacity is larger.

If an access is made to an address outside of the range defined by FSIZE but still within the 256-Mbyte range, then a bus error is given. The effect of this error depends on the bus master that attempted the access:

- If it is the Cortex CPU, bus fault exception is generated when enabled (or an HardFault exception when bus fault is disabled).
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

Byte, halfword, and word access types are all supported.

Support for execute in place (XIP) operation is implemented, where the QUADSPI anticipates the next access and load in advance the byte at the following address. If the subsequent access is indeed made at a continuous address, the access is completed faster since the value is already prefetched.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with NCS maintained low, even if no access to the flash memory occurs for a long time. Since flash memories tend to consume more when NCS is held low, the application may want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that NCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without any access since when the FIFO becomes full with prefetch data.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is a timeout, there is an abort, or the peripheral is disabled.

27.3.7 QUADSPI free-running clock mode

When configured in free-running clock mode, the QUADSPI continuously outputs the clock for test and calibration purposes.
The free-running clock mode is entered as soon as the FRCM bit is set in the QUADSPI_CCR register. It is exited by setting the ABORT bit of the QUADSPI_CR. When the QUADSPI operates in free-running clock mode:

- the clock is running continuously
- NCS stays high (external device deselected)
- data lines are released (High-Z)
- the BUSY flag of QUADSPI_SR is set

27.3.8 QUADSPI flash memory configuration

The device configuration register (QUADSPI_DCR) can be used to specify the characteristics of the external SPI flash memory.

The FSIZE[4:0] bitfield defines the size of external memory using the following formula:

$$\text{Number of bytes in flash memory} = 2^{[\text{FSIZE}+1]}$$

FSIZE+1 is effectively the number of address bits required to address the flash memory. The flash memory capacity can be up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DFM = 1, FSIZE indicates the total capacity of the two flash memories together.

When the QUADSPI executes two commands, one immediately after the other, it raises NCS high between the two commands for only one CLK cycle by default. If the flash memory requires more time between commands, the CSHT bitfield can be used to specify the minimum number of CLK cycles (up to 8) that NCS must remain high.

The clock mode (CKMODE) bit indicates the CLK signal logic level in between commands (when NCS =

1).

27.3.9 QUADSPI delayed data sampling

By default, the QUADSPI samples the data driven by the flash memory one half of a CLK cycle after the flash memory drives the signal.

In case of external signal delays, it may be beneficial to sample the data later. Using SSHIFT (bit 4 of QUADSPI_CR), the sampling of the data can be shifted by half of a CLK cycle.

Clock shifting is not supported in DDR mode: SSHIFT must be clear when DDRM bit is set.

QUADSPI configuration

The QUADSPI configuration is done in two phases:

1. QUADSPI peripheral configuration
2. QUADSPI flash memory configuration

Once configured and enabled, the QUADSPI can be used in one of its three operating modes: indirect, automatic status-polling, or memory-mapped mode.

QUADSPI configuration

The QUADSPI is configured using QUADSPI_CR. The user must configure the clock prescaler division factor and the sample shifting settings for the incoming data.

The DDR mode can be set through the DDRM bit. When setting the quad-SPI interface in DDR mode, the internal divider of kernel clock must be set with a division ratio of two or more. Once enabled, the address and the alternate bytes are sent on both clock edges, and the data are sent/received on both clock edges. Regardless of the DDRM bit setting, instructions are always sent in SDR mode.

The FIFO level for either MDMA trigger generation or interrupt generation is programmed in the FTHRES bits.

If a timeout counter is needed, the TCEN bit can be set and the timeout value programmed in the QUADSPI_LPTR register.

The dual-flash mode can be activated by setting DFM to 1.

QUADSPI flash memory configuration

The parameters related to the targeted external flash memory are configured through the QUADSPI_DCR register.The user must program the flash memory size in the FSIZE bits, the chip-select minimum high time in CSHT bits, and the functional mode (Mode 0 or Mode 3) in the MODE bit.

27.3.10 QUADSPI use

The operating mode is selected using FMODE[1:0] (QUADSPI_CCR[27:26]).

Indirect mode

When FMODE is programmed to 00, the indirect-write mode is selected and data can be sent to the flash memory. With FMODE = 01, the indirect-read mode is selected where data can be read from the flash memory.

When the QUADSPI is used in indirect mode, the frames are constructed in the following way:

- Specify a number of data bytes to read or write in QUADSPI_DLR.
- Specify the frame format, mode and instruction code in QUADSPI_CCR.
- Specify optional alternate byte to be sent right after the address phase in QUADSPI_ABR.
- Specify the operating mode in QUADSPI_CR.
- Specify the targeted address in QUADSPI_AR.
- Read/write the data from/to the FIFO through QUADSPI_DR.

When writing QUADSPI_CR, the user specifies the following settings:

- enable bit (EN) set to 1
- timeout counter enable bit (TCEN)
- sample shift setting (SSHIFT)
- FIFO threshold level (FTRHES) to indicate when the FTF flag must be set
- interrupt enables
- automatic status-polling mode parameters: match mode and stop mode (valid when FMODE = 11)
- clock prescaler

When writing QUADSPI_CCR, the user specifies the following parameters:

- instruction byte through INSTRUCTION bits
- the way the instruction has to be sent through the IMODE bits (1/2/4 lines)
- the way the address has to be sent through the ADMODE bits (None/1/2/4 lines)
- address size (8/16/24/32-bit) through ADSIZE bits
- the way the alternate bytes have to be sent through the ABMODE (None/1/2/4 lines)
- alternate bytes number (1/2/3/4) through the ABSIZE bits
- presence or not of dummy bytes through the DBMODE bit
- number of dummy bytes through the DCYC bits
- the way data have to be sent/received (none/1/2/4 lines) through DMODE bits

If neither QUADSPI_AR nor QUADSPI_DR need to be updated for a particular command, then the command sequence starts as soon as QUADSPI_CCR is written. This is the case when both ADMODE and DMODE are 00, or if just ADMODE = 00 when in indirect read mode (FMODE = 01).

When an address is required (ADMODE is not 00) and the data register does not need to be written (when FMODE = 01 or DMODE = 00), the command sequence starts as soon as the address is updated with a write to QUADSPI_AR.

In case of data transmission (FMODE = 00 and DMODE! = 00), the communication start is triggered by a write in the FIFO through QUADSPI_DR.

Automatic status-polling mode

This mode is enabled setting the FMODE bitfield (QUADSPI_CCR[27:26]) to 10. In this mode, the programmed frame is sent and the data retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in QUADSPI_DLR, it is ignored and only 4 bytes are read.

The periodicity is specified in the QUADSPI_PISR register.

Once the status data is retrieved, it can internally be processed:

- to set the status match flag and generate an interrupt if enabled
- to stop automatically the periodic retrieving of the status bytes

The received value can be masked with the value stored in QUADSPI_PSMKR and ORed or ANDed with the value stored in QUADSPI_PSMAR.

In case of match, the status match flag is set and an interrupt is generated if enabled, and the QUADSPI can be automatically stopped if the AMPS bit is set.

In any case, the latest retrieved value is available in QUADSPI_DR.

Memory-mapped mode

In memory-mapped mode, the external flash memory is seen as an internal memory but with some latency during accesses. Only read operations are allowed to the external flash memory in this mode.

The memory-mapped mode is entered by setting FMODE to 11 in QUADSPI_CCR.

The programmed instruction and frame is sent when a master is accessing the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate linear reads. Any access to QUADSPI_DR in this mode returns zero.

QUADSPI_DLR has no meaning in memory-mapped mode.

27.3.11 Sending the instruction only once

Some flash memories (for example: Winbound) provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. One can take advantage of such a feature using the SIOO bit (QUADSPI_CCR[28]).

SIOO is valid for all functional modes (indirect, automatic status-polling, and memory-mapped). If the SIOO bit is set, the instruction is sent only for the first command following a write to QUADSPI_CCR. Subsequent command sequences skip the instruction phase, until there is a write to QUADSPI_CCR.

SIOO has no effect when IMODE = 00 (no instruction).

27.3.12 QUADSPI error management

An error can be generated in the following case:

- In indirect mode or automatic status-polling mode when a wrong address is programmed in QUADSPI_AR (according to the flash memory size defined by FSIZE[4:0] in the QUADSPI_DCR), TEF is set and an interrupt is generated if enabled.
  Also in indirect mode, if the address plus the data length exceeds the flash memory size, TEF is set as soon as the access is triggered.
- In memory-mapped mode, when an out-of-range access is done by a master or when the QUADSPI is disabled, a bus error is generated as a response to the faulty bus master request.
  When a master is accessing the memory mapped space while the memory-mapped mode is disabled, a bus error is generated as a response to the faulty bus master request.

27.3.13 QUADSPI busy bit and abort functionality

Once the QUADSPI starts an operation with the flash memory, the BUSY bit is automatically set in QUADSPI_SR.

In indirect mode, BUSY is reset once the QUADSPI has completed the requested command sequence, and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1, or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on a timeout event or on an abort.

Any operation can be aborted by setting the ABORT bit in QUADSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

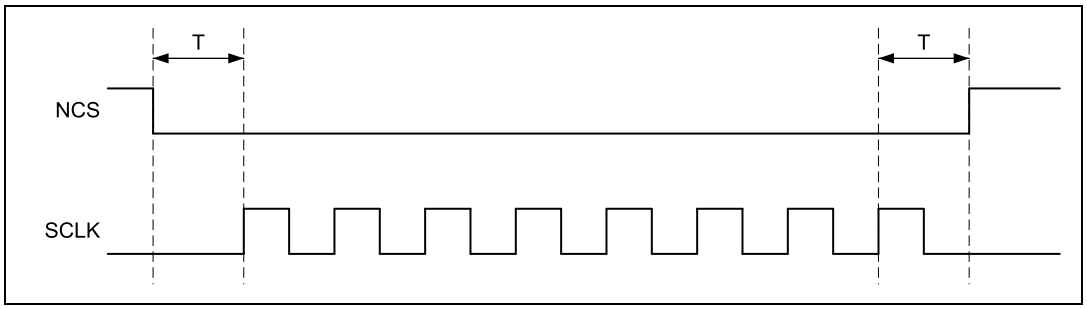
Note: *Some flash memories may misbehave if a write operation to a status registers is aborted.*

27.3.14 NCS behavior

By default, NCS is high, deselecting the external flash memory. NCS falls before an operation begins and rises as soon as it finishes.

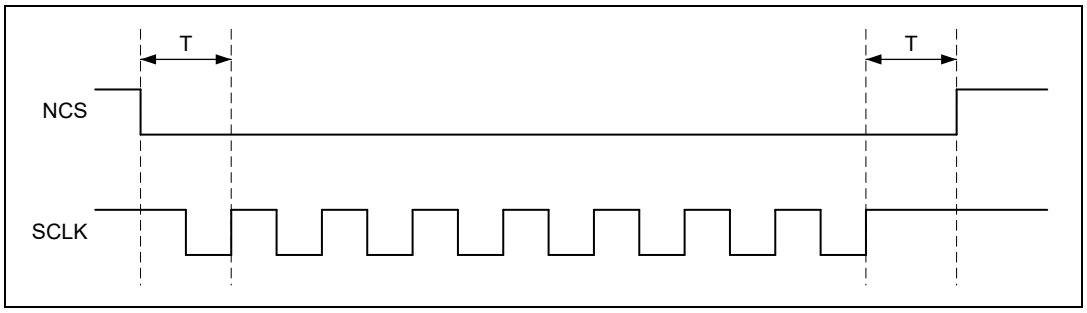
When CKMODE = 0 (“mode0”, where CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge, as shown in the figure below.

Fig 27.3-5 NCS when CKMODE = 0 (T = CLK period)



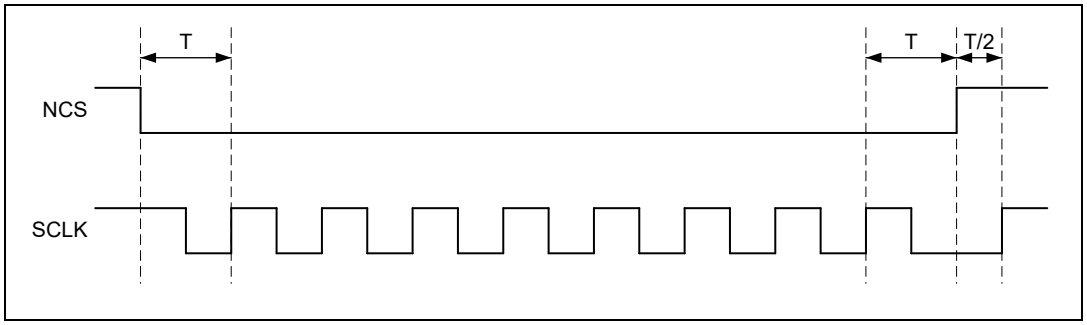
When CKMODE=1 (“mode3”, where CLK goes high when no operation is in progress) and DDRM=0 (SDR mode), NCS still falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge, as shown in the figure below.

Fig 27.3-6 NCS when CKMODE = 1 in SDR mode (T = CLK period)



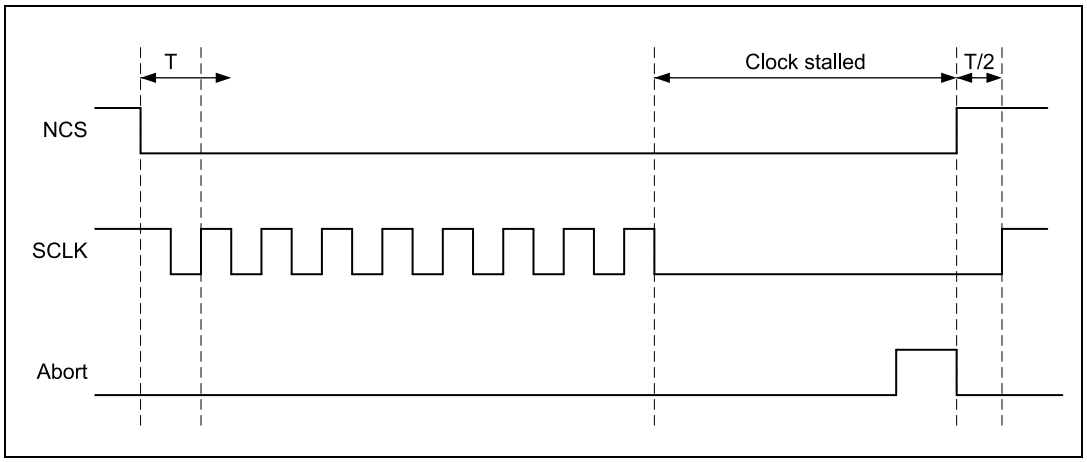
When CKMODE = 1 (“mode3”) and DDRM = 1 (DDR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge, as shown in the figure below. Because DDR operations must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Fig 27.3-7 NCS when CKMODE = 1 in DDR mode (T = CLK period)



When the FIFO stays full in a read operation or if the FIFO stays empty in a write operation, the operation stalls and CLK stays low until firmware services the FIFO. If an abort occurs when an operation is stalled, NCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later, as shown in 27.3-8

Fig 27.3-8 NCS when CKMODE = 1 with an abort (T = CLK period)



When not in dual-flash mode (DFM = 0) and FSEL = 0 (default value), only FLASH 1 is accessed. Thus BK2_NCS stays high, if FSEL = 1, only FLASH 2 is accessed and BK1_NCS stays high. In dual-flash mode, BK2_NCS behaves exactly the same as BK1_NCS. Thus, if there is a FLASH 2 and if the application is dual-flash mode only, then BK1_NCS signal can be used for FLASH 2 as well, and the pin devoted to BK2_NCS can be used for other functions.

27.3.15 QUADSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

Tab 27.3-3 QSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

27.4 QSPI Register

Tab 27.4-1 QSPI register map

offset	Register	Reset value	Description
QSPI_BA = 0x4001_4000			
0x000	QUADSPI_CR	0x0000_0000	QSPI control register
0x004	QUADSPI_DCR	0x0000_0000	QUADSPI device configuration register
0x008	QUADSPI_SR	0x0000_0000	QUADSPI status register
0x00C	QUADSPI_FCR	0x0000_0000	QUADSPI flag clear register
0x010	QUADSPI_DLR	0x0000_0000	QUADSPI data length register
0x014	QUADSPI_CCR	0x0000_0000	QUADSPI communication configuration register
0x018	QUADSPI_AR	0x0000_0000	QUADSPI address register
0x01C	QUADSPI_ABR	0x0000_0000	QUADSPI alternate-byte register
0x020	QUADSPI_DR	0x0000_0000	QUADSPI data register
0x024	QUADSPI_PSMKR	0x0000_0000	QUADSPI polling status mask register
0x028	QUADSPI_PSMAR	0x0000_0000	QUADSPI polling status match register
0x02C	QUADSPI_PIR	0x0000_0000	QUADSPI polling interval register
0x030	QUADSPI_LPTR	0x0000_0000	QUADSPI low-power timeout register

27.4.1 QUADSPI control register(QUADSPI_CR)

Register	Address offset	Access	Reset value	Description
QUADSPI_CR	0x00	RW	0x0000_0000	QUADSPI control register

31	30	29	28	27	26	25	24
PRESCALER							
23	22	21	20	19	18	17	16
PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
15	14	13	12	11	10	9	8
Reserved			FTHRES				
7	6	5	4	3	2	1	0
FSEL	DFM	Res.	SSHIFT	TCEN	DMEAN	ABORT	EN

QUADSPI control register(QUADSPI_CR) bit description

Bit	Access	Description
[31:24]	RW	**PRESCALER[7:0]:** Clock prescaler This bitfield defines the scaler factor for generating CLK based on the quadspi_ker_ck clock(value+1). 0: F_{CLK} = $F_{quadspi\_ker\_ck}$, quadspi_ker_ck clock used directly as QUADSPI CLK (prescaler by-passed) 1: F_{CLK} =$F_{quadspi\_ker\_ck}$ /2 2: F_{CLK}=$F_{quadspi\_ker\_ck}$ /3 ... 255:F_{CLK} =$F_{quadspi\_ker\_ck}$ /256 For odd clock division factors, CLK duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high. When setting quad-SPI interface in DDR mode, the prescaler must be set with a division ratio of two or more. **Note:** *This bitfield can be modified only when BUSY = 0.*
[23]	RW	**PMM:** Polling match mode This bit indicates which method must be used for determining a "match" during automatic status-polling mode. 0: AND match mode. SMF is set if all the unmasked bits received from the flash memory match the corresponding bits in the match register. 1: OR match mode. SMF is set if any one of the unmasked bits received from the flash memory matches its corresponding bit in the match register. **Note:** *This bit can be modified only when BUSY = 0.*
[22]	RW	**APMS:** Automatic status-polling mode stop This bit determines if automatic status-polling is stopped after a match. 0: Automatic status-polling mode is stopped only by abort or by disabling the QUADSPI. 1: Automatic status-polling mode stops as soon as there is a match. **Note:** *This bit can be modified only when BUSY = 0.*
[21]	R	Reserved, must be kept at reset value
[20]	RW	**TOIE:** Timeout interrupt enable This bit enables the timeout interrupt. 0: Interrupt disabled 1: Interrupt enabled

[19]	RW	**SMIE:** Status match interrupt enable This bit enables the status match interrupt. 0: Interrupt disabled 1: Interrupt enabled
[18]	RW	**FTIE:** FIFO threshold interrupt enable This bit enables the FIFO threshold interrupt. 0: Interrupt disabled 1: Interrupt enabled
[17]	RW	**TCIE:** Transfer complete interrupt enable This bit enables the transfer complete interrupt. 0: Interrupt disabled 1: Interrupt enabled
[16]	RW	**TEIE:** Transfer error interrupt enable This bit enables the transfer error interrupt. 0: Interrupt disabled 1: Interrupt enabled
[15:13]	R	Reserved , must be kept at reset value
[12:8]	RW	**FTHRES[4:0]:** FIFO threshold level This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag (bit FTF in register QUADSPI_SR) to be set. 0: In indirect-write mode (FMODE = 00), FTF is set if there are one or more free bytes location left in the FIF,O or indirect-read mode (FMODE = 01), FTF is set if there are one or more valid bytes that can be read from the FIFO. 1: In indirect-write mode (FMODE = 00), FTF is set if there are two or more free bytes location left in the FIFO, or indirect-read mode (FMODE = 01), FTF is set if there are two or more valid bytes that can be read from the FIFO ... 31: In indirect-write mode (FMODE = 00), FTF is set if there are 32 free bytes location left in the FIFO, or indirect-read mode (FMODE = 01), FTF is set if there are 32 valid bytes that can be read from the FIFO.
[7]	RW	**FSEL:** Flash memory selection This bit selects the flash memory to be addressed in single-flash mode (when DFM = 0). 0: FLASH 1 selected 1: FLASH 2 selected **Note:** *This bit can be modified only when BUSY = 0. This bit is ignored when DFM = 1.*
[6]	RW	**DFM:** Dual-flash mode This bit activates dual-flash mode, where two external flash memories are used simultaneously to double throughput and capacity. 0: Dual-flash mode disabled 1: Dual-flash mode enabled **Note:** *This bit can be modified only when BUSY = 0*
[5]	R	Reserved , must be kept at reset value

[4]	RW	**SSHIFT:** Sample shift By default, the QUADSPI samples data 1/2 of a CLK cycle after the data is driven by the flash memory. This bit allows the data to be sampled later in order to account for external signal delays. 0: No shift 1: 1/2 cycle shift The firmware must assure that SSHIFT = 0 when in DDR mode (when DDRM = 1). **Note:** *This bitfield can be modified only when BUSY = 0.*
[3]	RW	**TCEN:** Timeout counter enable This bit is valid only when memory-mapped mode (FMODE = 11) is selected. Activating this bit causes the NCS to be released (and thus reduces consumption) if there has not been an access after a certain amount of time, where this time is defined by TIMEOUT[15:0] (QUADSPI_LPTR). This bit enables the timeout counter. By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with NCS maintained low, even if no access to the flash memory occurs for a long time. Since flash memories tend to consume more when NCS is held low, the application may want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that NCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without an access since when the FIFO becomes full with prefetch data. 0: Timeout counter is disabled, and thus the NCS remains active indefinitely after an access in memory-mapped mode. 1: Timeout counter is enabled, and thus the NCS is released in memory-mapped mode after TIMEOUT[15:0] cycles of flash memory inactivity. **Note:** *This bit can be modified only when BUSY = 0*
[2]	R	Reserved , must be kept at reset value
[1]	RW	**ABORT:** Abort request This bit aborts the ongoing command sequence. It is automatically reset once the abort is complete. This bit stops the current transfer. In automatic status-polling or memory-mapped mode, this bit also reset APM or DM bit. 0: No abort requested 1: Abort requested
[0]	RW	**EN:** QUADSPI enable 0: QUADSPI disabled 1: QUADSPI enabled

27.4.2 QUADSPI device configuration register(QUADSPI_DCR)

Register	Address offset	Access	Reset value	Description
QUADSPI_DCR	0x004	RW	0x0000_0000	QUADSPI device configuration register

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

QUADSPI device configuration register(QUADSPI_DCR) bit description

Bit	Access	Description
[31:21]	R	Reserved, must be kept at reset value.
[20:16]	RW	**FSIZE[4:0]:** Flash memory size This bitfield defines the size of external memory using the following formula: Number of bytes in flash memory = $2^{[FSIZE+1]}$ FSIZE+1 is effectively the number of address bits required to address the flash memory. The flash memory capacity can be up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes. If DFM = 1, FSIZE indicates the total capacity of the two flash memories together. **Note:** *This bitfield can be modified only when BUSY = 0.*
[15:11]	R	Reserved , must be kept at reset value
[10:8]	RW	**CSHT[2:0]:** Chip select high time CSHT+1 defines the minimum number of CLK cycles which the chip select (NCS) must remain high between commands issued to the flash memory. 0: NCS stays high for at least 1 cycle between flash memory commands 1: NCS stays high for at least 2 cycles between flash memory commands ... 7: NCS stays high for at least 8 cycles between flash memory commands **Note:** *This bitfield can be modified only when BUSY = 0.*
[7:1]	R	Reserved , must be kept at reset value
[0]	RW	**CKMODE:** Mode 0/mode 3 This bit indicates the level that CLK takes between commands (when NCS = 1). 0: CLK must stay low while NCS is high (chip select released). This is referred to as mode 0. 1: CLK must stay high while NCS is high (chip select released). This is referred to as mode 3. **Note:** *This bitfield can be modified only when BUSY = 0*

27.4.3 QUADSPI status register(QUADSPI_SR)

Register	Address offset	Access	Reset value	Description
QUADSPI_SR	0x008	RW	0x0000_0000	QUADSPI status register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		FLEVEL[5:0]					
7	6	5	4	3	2	1	0
Reserved		BUSY	TOF	SMF	FTF	TCF	TEF

QUADSPI status register(QUADSPI_SR) bit description

Bit	Access	Description
[31:14]	R	Reserved, must be kept at reset value.
[13:8]	RW	**FLEVEL[5:0]:** FIFO level This bitfield gives the number of valid bytes which are being held in the FIFO.FLEVEL = 0 when the FIFO is empty, and 32 when it is full. In memory-mapped mode and in automatic status-polling mode, FLEVEL is zero.

[7:6]	R	Reserved , must be kept at reset value
[5]	RW	**BUSY:** Busy This bit is set when an operation is on going. This bit clears automatically when the operation with the flash memory is finished and the FIFO is empty.
[4]	RW	**TOF:** Timeout flag This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.
[3]	RW	**SMF:** Status match flag This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (QUADSPI_PSMAR). It is cleared by writing 1 to CSMF.
[2]	RW	**FTF:** FIFO threshold flag In indirect mode, this bit is set when the FIFO threshold is reached, or if there is any data left in the FIFO after reads from the flash memory are complete. It is cleared automatically as soon as threshold condition is no longer true. In automatic status-polling mode this bit is set every time the status register is read, and the bit is cleared when the data register is read.
[1]	RW	**TCF:** Transfer complete flag This bit is set in indirect mode when the programmed number of data is transferred or in any mode when the transfer is aborted.It is cleared by writing 1 to CTCF.
[0]	RW	**TEF:** Transfer error flag This bit is set in indirect mode when an invalid address is being accessed in indirect mode. It is cleared by writing 1 to CTEF.

27.4.4 QUADSPI flag clear register(QUADSPI_FCR)

Register	Address offset	Access	Reset value	Description
QUADSPI_FCR	0x00C	RW	0x0000_0000	QUADSPI flag clear register

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

QUADSPI flag clear register(QUADSPI_FCR) bit description

Bit	Access	Description
[31:5]	R	Reserved, must be kept at reset value.
[4]	RW	**CTOF:** Clear timeout flag Writing 1 clears the TOF flag in QUADSPI_SR
[3]	RW	**CSMF:** Clear status match flag Writing 1 clears the SMF flag in QUADSPI_SR
[2]	R	Reserved, must be kept at reset value.
[1]	RW	**CTCF:** Clear transfer complete flag Writing 1 clears the TCF flag in QUADSPI_SR.

[0]	RW	**CTEF:** Clear transfer error flag Writing 1 clears the TEF flag in QUADSPI_SR

27.4.5 QUADSPI data length register (QUADSPI_DLR)

Register	Address offset	Access	Reset value	Description
QUADSPI_DLR	0x010	RW	0x0000_0000	QUADSPI data length register

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

QUADSPI data length register(QUADSPI_DLR) bit description

Bit	Access	Description
[31:0]	RW	**DL[31:0]:** Data length Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value no greater than 3 (indicating 4 bytes) must be used for automatic status-polling mode. All 1s in indirect mode means undefined length, where the QUADSPI continues until the end of memory, as defined by FSIZE. 0x0000_0000: 1 byte is to be transferred 0x0000_0001: 2 bytes are to be transferred 0x0000_0002: 3 bytes are to be transferred 0x0000_0003: 4 bytes are to be transferred ... 0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred 0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred 0xFFFF_FFFF: undefined length – all bytes until the end of flash memory (as defined by FSIZE) are to be transferred. Continue reading indefinitely if FSIZE = 0x1F. DL[0] is stuck at 1 in dual-flash mode (DFM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes. This bitfield has no effect when in memory-mapped mode (FMODE = 10). **Note:** *This bitfield can be written only when BUSY = 0.*

27.4.6 QUADSPI communication configuration register (QUADSPI_CCR)

Register	Address offset	Access	Reset value	Description
QUADSPI_CCR	0x014	RW	0x0000_0000	QUADSPI communication configuration register

31	30	29	28	27	26	25	24
Reserved			SIOO	FMODE[1:0]		DMODE[1:0]	
23	22	21	20	19	18	17	16
Reserved	DCYC[4:0]					ABSIZE[1:0]	
15	14	13	12	11	10	9	8
ABMODE[1:0]		ADSIZE[1:0]		ADMODE[1:0]		IMODE[1:0]	
7	6	5	4	3	2	1	0
INSTRUCTION[7:0]							

QUADSPI communication configuration register (QUADSPI_CCR) bit description

Bit	Access	Description
[31:29]	R	Reserved
[28]	RW	**SIOO:** Send instruction only once mode This bit has no effect when IMODE = 00.See Section 27.3.11 for more details. 0: Instruction sent on every transaction 1: Instruction sent only for the first command **Note:** *This bit can be written only when BUSY = 0.*
[27:26]	RW	**FMODE[1:0]:** Functional mode This bitfield defines the QUADSPI functional mode of operation. 00: Indirect-write mode 01: Indirect-read mode 10: Automatic status-polling mode 11: Memory-mapped mode **Note:** *This bitfield can be written only when BUSY = 0*
[25:24]	RW	**DMODE[1:0]:** Data mode This bitfield defines the data phase mode of operation: 00: No data 01: Data on a single line 10: Data on two lines 11: Data on four lines This bitfield also determines the dummy phase mode of operation. **Note:** *This bitfield can be written only when BUSY = 0.*
[23]	R	Reserved, must be kept at reset value
[22:18]	RW	**DCYC[4:0]:** Number of dummy cycles This bitfield defines the duration of the dummy phase. In both SDR and DDR modes, it specifies a number of CLK cycles (0-31). **Note:** *This bitfield can be written only when BUSY = 0.*
[17:16]	RW	**ABSIZE[1:0]:** Alternate-byte size This bit defines the size of alternate bytes. 00: 8-bit alternate byte 01: 16-bit alternate bytes 10: 24-bit alternate bytes 11: 32-bit alternate bytes **Note:** *This bitfield can be written only when BUSY = 0.*
[15:14]	RW	**ABMODE[1:0]:** Alternate byte mode This bitfield defines the alternate-byte phase mode of operation. 00: No alternate bytes 01: Alternate bytes on a single line 10: Alternate bytes on two lines 11: Alternate bytes on four lines **Note:** *This bitfield can be written only when BUSY = 0.*

[13:12]	RW	**ADSIZE[1:0]:** Address size This bit defines address size: 00: 8-bit address 01: 16-bit address 10: 24-bit address 11: 32-bit address **Note:** *This bitfield can be written only when BUSY = 0.*
[11:10]	RW	**ADMODE[1:0]:** Address mode This bitfield defines the address phase mode of operation. 00: No address 01: Address on a single line 10: Address on two lines 11: Address on four lines **Note:** *This bitfield can be written only when BUSY = 0.*
[9:8]	RW	**IMODE[1:0]:** Instruction mode This bitfield defines the instruction phase mode of operation. 00: No instruction 01: Instruction on a single line 10: Instruction on two lines 11: Instruction on four lines **Note:** *This bitfield can be written only when BUSY = 0.*
[7:0]	RW	**INSTRUCTION[7:0]:** Instruction Instruction to be sent to the external SPI device. **Note:** *This bitfield can be written only when BUSY = 0.*

27.4.7 QUADSPI address register (QUADSPI_AR)

Register	Address offset	Access	Reset value	Description
QUADSPI_AR	0x018	RW	0x0000_0000	QUADSPI address register

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

QUADSPI address register (QUADSPI_AR)bit description

Bit	Access	Description
[31:0]	RW	**ADDRESS[31:0]:** Address This bitfield contains the address to be sent to the external flash memory. Writes to this bitfield are ignored when BUSY = 1 or when FMODE = 11 (memory-mapped mode). In dual flash mode, ADDRESS[0] is automatically stuck to 0 as the address must always be even

27.4.8 QUADSPI alternate-byte register (QUADSPI_ABR)

Register	Address offset	Access	Reset value	Description
QUADSPI_ABR	0x01C	RW	0x0000_0000	QUADSPI alternate-byte register

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

QUADSPI alternate-byte register (QUADSPI_ABR)bit description

Bit	Access	Description
[31:0]	RW	**ALTERNATE[31:0]:** Alternate bytes Optional data to be send to the external SPI device right after the address. Note: This bitfield can be written only when BUSY = 0

27.4.9 QUADSPI data register (QUADSPI_DR)

Register	Address offset	Access	Reset value	Description
QUADSPI_DR	0x020	RW	0x0000_0000	QUADSPI data register

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]															

QUADSPI data register (QUADSPI_DR)bit description

Bit	Access	Description
[31:0]	RW	**DATA[31:0]:** Data Data to be sent/received to/from the external SPI device. In indirect write mode, data written to this register is stored on the FIFO before it is sent to the flash memory during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written. In indirect read mode, reading this register gives (via the FIFO) the data which was received from the flash memory. If the FIFO does not have as many bytes as requested by the read operation and if BUSY=1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first. In automatic status-polling mode, this register contains the last data read from the flash memory (without masking). Word, halfword, and byte accesses to this register are supported. In indirect write mode, a byte write adds 1 byte to the FIFO, a halfword write 2, and a word write 4. Similarly, in indirect read mode, a byte read removes 1 byte from the FIFO, a halfword read 2, and a word read 4. Accesses in indirect mode must be aligned to the bottom of this register: a byte read must read DATA[7:0] and a halfword read must read DATA[15:0].

27.4.10 QUADSPI polling status mask register (QUADSPI_PSMKR)

Register	Address offset	Access	Reset value	Description
QUADSPI_PSMKR	0x024	RW	0x0000_0000	QUADSPI polling status mask register

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

QUADSPI polling status mask register (QUADSPI_PSMKR)bit description

Bit	Access	Description
[31:0]	RW	**MASK[31:0]:** Status mask Mask to be applied to the status bytes received in automatic status-polling mode. For bit n: 0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic 1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic **Note:** *This bitfield can be written only when BUSY = 0.*

27.4.11 QUADSPI polling status match register (QUADSPI_PSMAR)

Register	Address offset	Access	Reset value	Description
QUADSPI_PSMAR	0x028	RW	0x0000_0000	QUADSPI polling status match register

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

QUADSPI polling status match register(QUADSPI_PSMAR)bit description

Bit	Access	Description
[31:0]	RW	**MATCH[31:0]:** Status match Value to be compared with the masked status register to get a match. **Note:** *This bitfield can be written only when BUSY = 0.*

27.4.12 QUADSPI polling interval register(QUADSPI_PIR)

Register	Address offset	Access	Reset value	Description
QUADSPI_PIR	0x02C	RW	0x0000_0000	QUADSPI polling interval register

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

QUADSPI polling interval register(QUADSPI_PIR)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.
[15:0]	RW	**INTERVAL[15:0]:** Polling interval Number of CLK cycles between two read during automatic status-polling phases. **Note:** *This bitfield can be written only when BUSY = 0.*

27.4.13 QUADSPI low-power timeout register(QUADSPI_LPTR)

Register	Address offset	Access	Reset value	Description
QUADSPI_LPTR	0x030	RW	0x0000_0000	QUADSPI low-power timeout register

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

QUADSPI low-power timeout register(QUADSPI_LPTR)bit description

Bit	Access	Description
[31:16]	R	Reserved, must be kept at reset value.
[15:0]	RW	**TIMEOUT[15:0]:** Timeout period After each access in memory-mapped mode, the QUADSPI prefetches the subsequent bytes and holds these bytes in the FIFO. This bitfield indicates how many CLK cycles the QUADSPI waits after the FIFO becomes full until it raises NCS, putting the flash memory in a lower-consumption state. **Note:** *This bitfield can be written only when BUSY = 0.*

28 Operational Amplifier (OPA)

28.1 OPA Function description

The product integrates three separate rail-to-rail high-speed operational amplifiers. The three I/ OS of each rail-to-rail high-speed operational amplifier can be connected to the external pins, allowing any type of external interconnection through the external circuit, while supporting the output to be connected to the internal ADC input channel.

Rail to rail input/output voltage range (0 VDDA)

Low input offset voltage (less than 5mV)

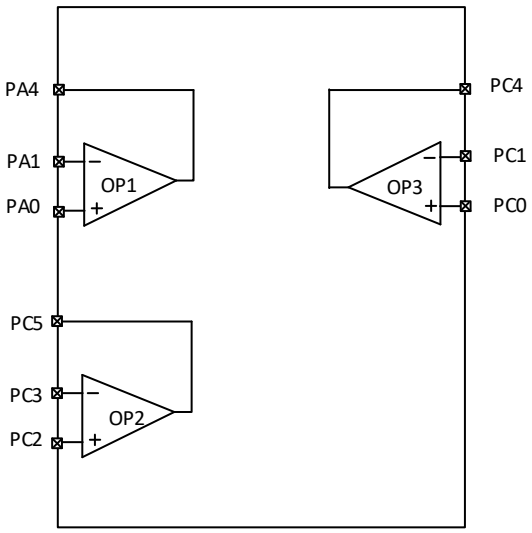
Rapid establishment time (0.1μs)

High open loop gain (106dB)

High power supply rejection ratio (103dB)

Gain bandwidth 20MHz (Cload=100pF)

Fig 28.1-1 OPA pin layout diagram



28.2 OPA Register

28.2.1 OPAMP1 Control/status register(OPAx_CSR,x=1,2,3)

Register	Address offset	Access	Reset value	Description
OPAx_CSR(x=1,2,3)	0x00,0x80,0x100	RW	0x0000_0000	OPAMP1 Control/status register

31	30	29	28	27	26	25	24	23	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															OPAEN

OPAMP1 Control/status register(OPAx_CSR,x=1,2,3)bit description

Bit	Access	Description
[31:1]	R	Reserved, must be kept at reset value.
[0]	RW	**OPAEN:** Operational amplifier enabled 0: operational amplifier disabled 1: Operational amplifier is enabled

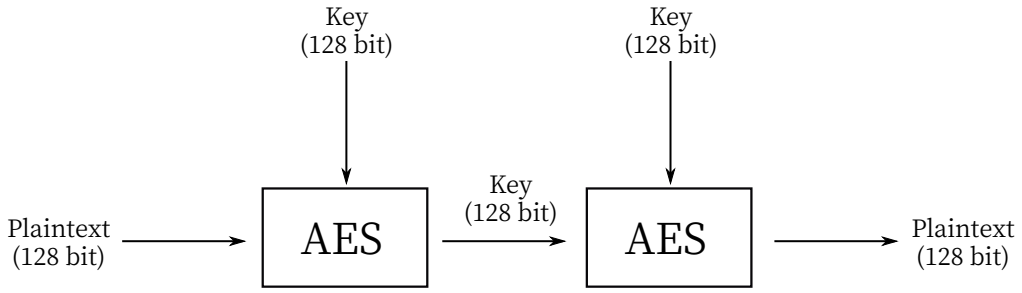
29 AES hardware accelerator

29.1 AES algorithm

AES is the full name of Advanced Encryption Standard, also known as Rijndael encryption, is a block encryption standard adopted by the federal government of the United States.

To put it simply, AES is used to encrypt the 128bit plaintext with a 128bit (can try 192bit or 256bit) key, and then obtain a 128bit ciphertext. After receiving the ciphertext, the receiver decrypts the ciphertext with the same key to obtain the plaintext.

AES has many working modes, mainly for data grouping and linking some differences, these modes include ECB, CBC, CTR, CFB, OFB, etc.



29.2 AES Main characteristics

- In full compliance with NIST FIPS PUB 197 standards;
- supports only 128bit keys.
- supports key derivation;
- Support ECB/CBC/CTR link mode;
- Support DMA interface (send and receive two independent channels);
- 220 clocks are required to complete an AES operation;
- Supports error detection
- Interrupt support: Processing ends and an error occurs

29.3 AES Function description

The AES hardware accelerator has four main operating modes, which can be selected according to the MODE register:

Mode1: Encryption, using the value of the key register to encrypt the data in the input cache, and the result is saved in the output cache;
Mode2: key derivation, the key register value derived calculation, the results continue to save in the key register;
Mode3: Decryption, using the value of the key register to decrypt the data in the input cache,

and the result is saved in the output cache;
Mode4: Key derivation + decryption, derive the value of the key register and then use it to decrypt the input (not available in CTR mode);

AES hardware accelerator supports chaining processing, that is, continuously input data in the fill phase (when CCF is low) after EN is raised. Note that the key, IV, and control registers cannot be changed after EN is raised. There are three CHMOD modes to choose from:

ECB: No IV is required, each block uses a fixed key;
CBC: Requires IV (fill the initial vector), uses IV for the first block, and uses the output of the previous block for each subsequent block calculation;
CTR: IV (fill the initial value of the counter) is required. The calculation of each block requires the value of the counter, which is added by one after each use;

CCF is only lifted up after the end of calculation, and then values in data output cache (mode 1/3/4) or key register (mode 2) are effective. Setting CCFIE=1 can generate interruption when CCF is lifted up. Write errors occur when:

- Set CHMOD, MODE, DATATYPE, IV, and KEY when EN is raised.
- Configure CHMOD, MODE, and DATATYPE during the calculation.
- Fill operation when the data of the current block in the input cache is not taken from the calculation;

Read errors occur in the following cases:

- Read when the current block is not filled in the output cache;

Key, input, output, and IV are all small-endian formats, which can be adjusted through DATATYPE.

29.4 Key derivation

Four key registers are required for key derivation, and the result is written back to these four registers after derivation.

Of the four key registers, KEYR0 is the lowest 32 bit of the whole 128-bit key, and KEYR3 is the highest 32 bit;

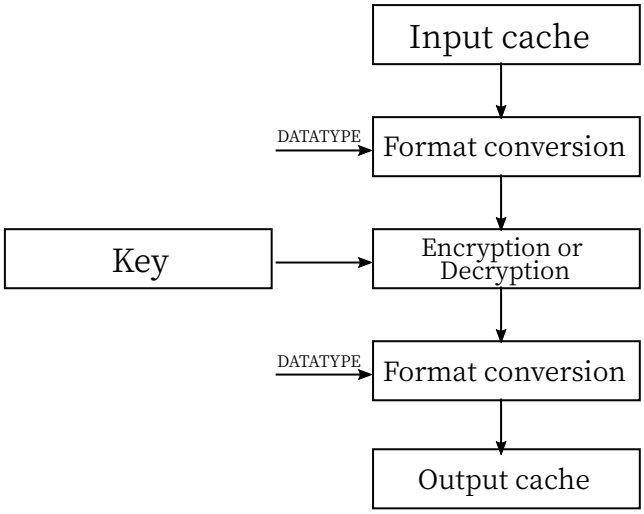
Because KEY can only be configured when EN is low, when MODE=2 or 4, the hardware automatically starts the derived calculation by pulling EN high;

After derivation of MODE=2, you can lower EN and modify MODE=3 to start decryption. At this time, you do not need to configure the key, or you can directly use MODE=4 to complete the two steps.

29.5 AES link mode

29.5.1 ECB model

This mode does not require IV, so the processing of each block is actually independent of each other



29.5.2 CBC mode

In this mode, each block will use the calculation result of the previous block, which has strong inheritance. The first block will use the register configuration IV (initial vector).

Fig 29.5-1 Encryption process

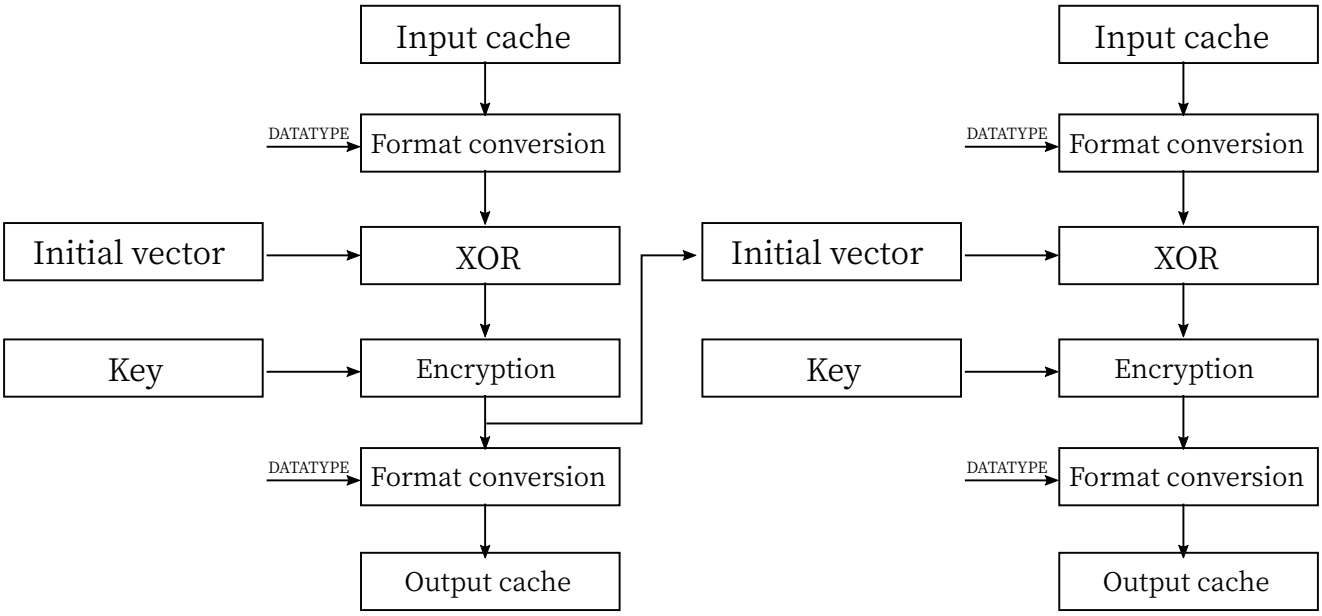
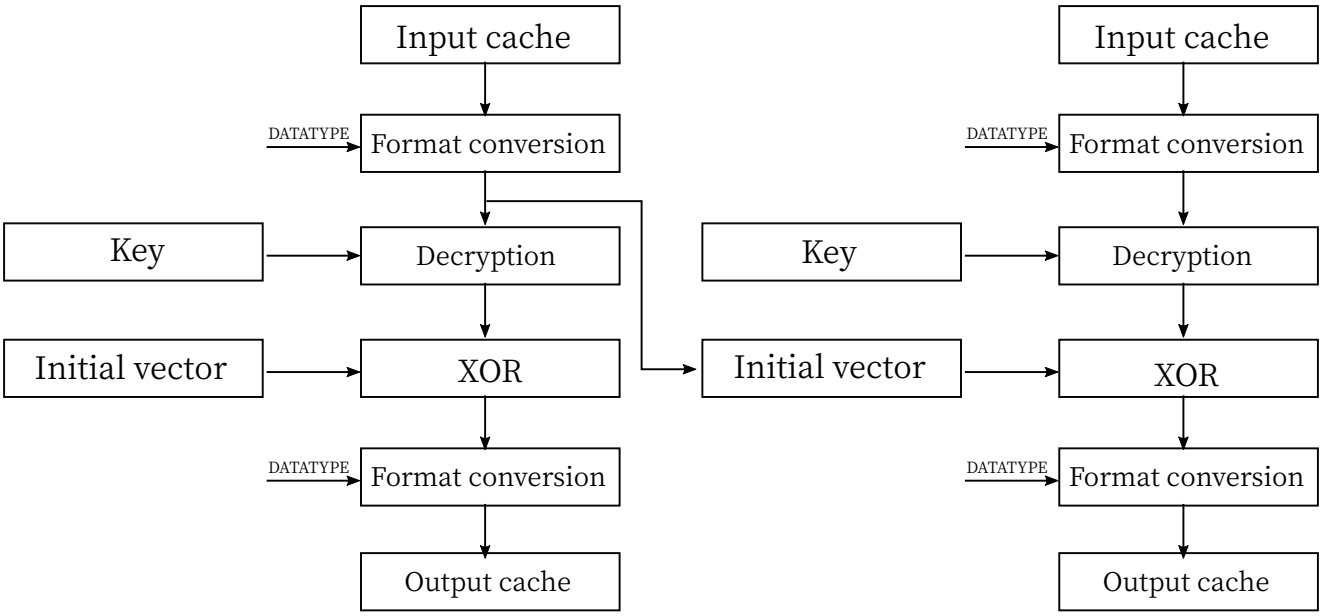
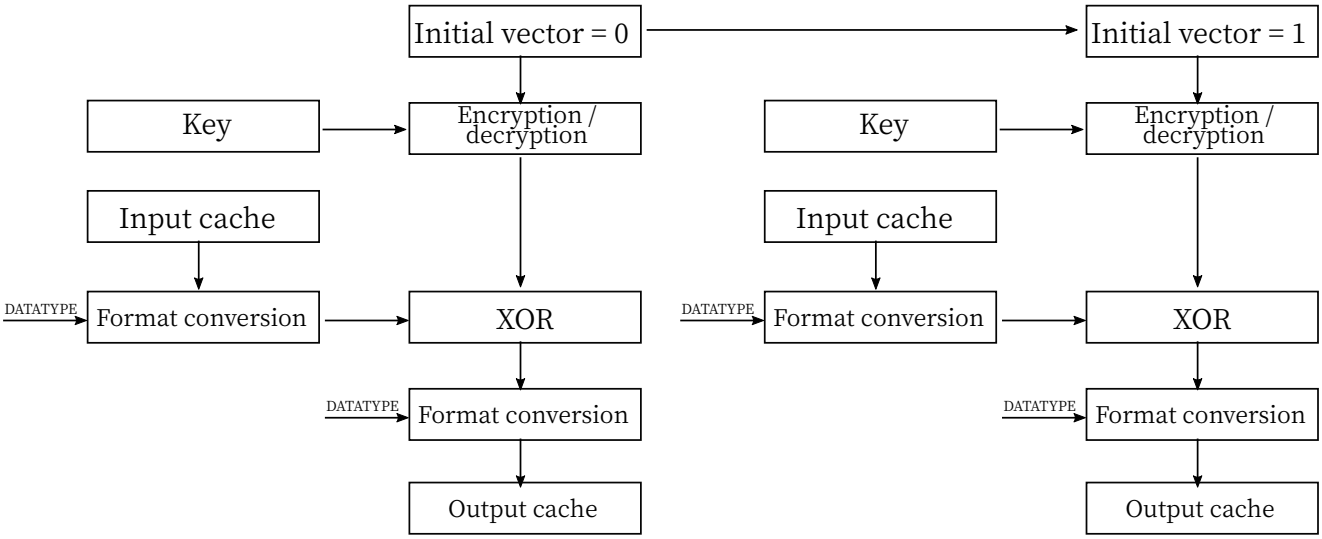


Fig 29.5-2 Decryption process



29.5.3 CTR mode

This pattern uses a counter to maintain the continuity of each block



29.5.4 Pause function

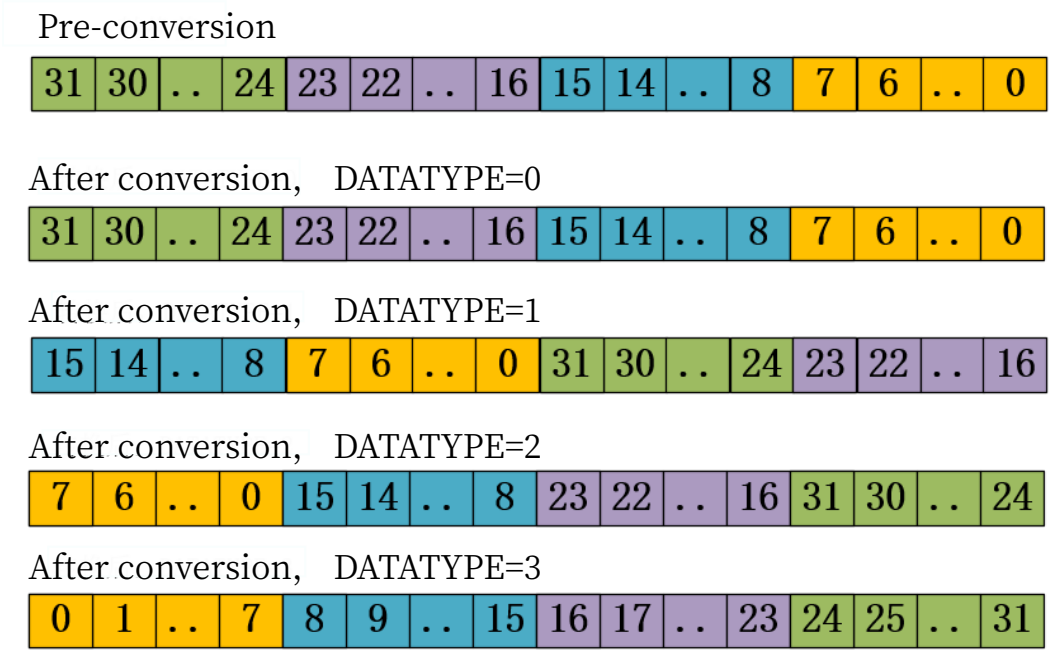
In practice, the AES encryption and decryption flow may be required for important packet B. In this case, you can encrypt and decrypt the current packet A Process suspension, the specific process is:

1. The N-1 block of packet A ends. Procedure
2. The NTH block of packet A ends. Procedure
3. If an interrupt is received, packet B needs to be processed

4. Read the current IVR value xxx and store it in SRAM
5. The processing of packet B starts
6. End of Message B
7. The IVR retrieves the N+1 block of packet A from the SRAM

29.5.5 Data format conversion

Data in and out of the cache will pass through the 32-bit register. There will be a format conversion when the register fills the input cache and when it is taken out of the output cache and written to the register. Through the register DATATYPE selection:



29.6 Use process

29.6.1 Procedure for using Mode1(encryption)

1. Lower the EN
2. Set MODE to 0 and select CHMOD
3. Configure the key
4. Configure IV (CBC and CTR modes).
5. Pull up the EN
6. Fill the maximum 32 bits of data into the input register
7. continue to fill 32 bits of data into the input register
8. Continue to fill 32 bits of data into the input register
9. Fill the lowest 32 bits of data into the input register
10. Wait for CCF to raise (interrupt or poll)
11. Read from the output register, this time corresponding to the highest 32 bits
12. Continue reading from the output register
13. Continue reading from the output register

14. read from the output register, this time corresponding to the lowest 32 bits
15. Repeat Steps 6-14

29.6.2 Use flow of Mode2(key derivation)

1. Lower the EN
2. Set MODE to 1
3. Configure the key
4. Pull up the EN
5. Wait for CCF to pull up (interrupt or poll)
6. Read the key

29.6.3 Use of Mode3(Decryption)

1. Lower the EN
2. Set MODE to 2 and select CHMOD
3. Configure the key
4. Configure IV (CBC and CTR modes).
5. Pull up the EN
6. Fill up to 32 bits of data into the input register
7. Continue to fill 32 bits of data into the input register
8. Continue to fill 32 bits of data into the input register
9. Fill the lowest 32 bits of data into the input register
10. Wait for CCF to raise (interrupt or poll)
11. read from the output register, this time corresponding to the highest 32 bits
12. Continue reading from the output register
13. Continue reading from the output register
14. read from the output register, this time corresponding to the lowest 32 bits
15. Repeat Steps 6-14

29.6.4 Mode4(Derivation + Decryption) use flow

1. Lower the EN
2. Set MODE to 3 and select CHMOD
3. Configure the key
4. Configure IV (CBC and CTR modes).
5. Pull up the EN
6. Fill the maximum 32 bits of data into the input register
7. Continue to fill 32 bits of data into the input register
8. Continue to fill 32 bits of data into the input register
9. Fill the lowest 32 bits of data into the input register
10. Wait for CCF to raise (interrupt or poll)
11. Read from the output register, this time corresponding to the highest 32 bits
12. Continue reading from the output register
13. Continue reading from the output register

14. read from the output register, this time corresponding to the lowest 32 bits
15. Repeat Steps 6-14

29.7 DMA interface

The AES hardware accelerator contains two DMA interfaces, one for data input and one for output.

Input interface workflow:

1. DMAINEN is set to 1;
2. Set other parameters.
3. EN pull up;
4. Send a request;
5. Receive data written to input cache (up to 32 bits);
6. Send a request;
7. Continue to receive data written to the cache;
8. Send a request;
9. Continue to receive data written to the cache;
10. Send a request;
11. Write the received data to the cache (minimum 32 bits);
12. Stop the request until this AES block processing is complete (four 32-bit outputs are taken);
13. Restart the request

Output interface workflow:

1. DMAOUTEN is set to 1;
2. Set other parameters.
3. EN pull up;
4. Wait for AES calculation to complete, output cache update;
5. Send a request;
6. Output cache data (up to 32 bits) is fetched;
7. Send a request;
8. Output cache data continues to be fetched;
9. Send a request;
10. Output cache data continues to be fetched;
11. Send a request;
12. Output cache data (minimum 32 bits) is removed;
13. Stop the request and wait for the next output cache update;
14. Restart the request;

29.8 AES Register

Tab 29.8-1 AES register map

Offset	Register	Reset value	Description
AES_BA = 0x4002_6000			
0x00	AES_CR	0x0000_0000	AES control register
0x04	AES_SR	0x0000_0000	AES Status register
0x08	AES_DINR	0x0000_0000	AES data input register
0x0C	AES_OUTR	0x0000_0000	AES data output register
0x10	AES_KEYR0(key[31:0])	0x0000_0000	AES key register0
0x14	AES_KEYR1(LSB:key[63	0x0000_0000	AES key register1
0x18	AES_KEYR2(key[95:64]	0x0000_0000	AES key register2
0x1C	AES_KEYR3(key[127:96	0x0000_0000	AES key register3
0x20	AES_IVR0(IVR[31:0])	0x0000_0000	AES Initializes the vector register0
0x24	AES_IVR1(IVR[63:32])	0x0000_0000	AES Initializes the vector register1
0x28	AES_IVR2(IVR[95:64])	0x0000_0000	AES Initializes the vector register2
0x2C	AES_IVR3(IVR[127:96])	0x0000_0000	AES Initializes the vector register3

29.9 AES Registers

29.9.1 AES control register(AES_CR)

Register	Address offset	Access	Reset value	Description
AES_CR	0x00	RW	0x0000_0000	AES control register

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC
7	6	5	4	3	2	1	0
CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN

AES control register(AES_CR)bit description

Bit	Access	Description
[31:13]	R	Reserved
[12]	RW	**DMAOUTEN:** DMA interface to enable data output caching: 0: DMA disabled (during data output phase) 1: DMA enabled (during data output phase) If the DMAOUTEN bit is set, a DMA request is generated for the output data phase in modes 1, 3, or 4. This bit is not valid in mode 2 (key derivation)
[11]	RW	**DMAINEN:** DMA interface to enable data input caching: 0: DMA disabled (during data input phase) 1: DMA enabled (in data input phase) If the DMAINEN bit is set, the DMA request is generated for the data entry phase in modes 1, 3, or 4. This bit has no effect in mode 2 (key derivation).
[10]	RW	**ERRIE:** Error interrupt was enabled If at least one of the RDERR or WRERR flags is set, an interrupt is generated. 0: Error interrupt disabled 1: Error interrupt is enabled
[9]	RW	**CCFIE:** CCF Indicates that interrupt is enabled If the CCF flag is set, an interrupt is generated. 0: disables CCF interrupt 1: indicates that CCF interrupt is enabled
[8]	RW	**ERRC:** Error clearing Writing 1 to this bit clears the RDERR and WRERR flags. This bit is always read as low
[7]	RW	**CCFC:** The calculation completed flag is cleared Writing a 1 to this bit clears the CCF flag. This bit is always read as low.
[6:5]	RW	**CHMOD[1:0]:** AES link mode 00: Electronic Codebook (EBC) 01: Cipher Block Chain (CBC) 10: Counter mode (CTR) 11: Reserve. This can only be changed when EN=0

[4:3]	RW	**MODE[1:0]:** AES operating mode 00: Mode 1: encryption 01: Mode 2: key derivation 10: Mode 3: Decryption 11: Mode 4: Key derivation + decryption This can only be changed when EN=0
[2:1]	RW	**DATATYPE[1:0]:** Data type selection (for encrypting block data input and data output) 00: no exchange. 01: 16 bit switch, 0x764356AB is converted to 0x56AB7643 10: 8 bit switch, 0x764356AB converted to 0xAB564376. 11: 1 bit switch, 0x764356AB is converted to 0xD56AC26E This can only be changed when EN=0
[0]	RW	**EN:** AES enablement 0: AES is disabled 1: AES is enabled AES can be reinitialized at any time by resetting this bit: AES is then ready to start processing the new block when EN is set. This bit is cleared by the hardware when the AES calculation is completed in Mode 2 (key derivation).

29.9.2 AES Status register(AES_SR)

Register	Address offset	Access	Reset value	Description
AES_SR	0x04	R	0x0000_0000	AES Status register

31	30	29	28	27	26	25	24	23	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													WRERR	RDERR	CCF

AES control register(AES_CR)bit description

Bit	Access	Description
[31:3]	R	Reserved
[2]	R	**WRERR:** Write error flag This bit is set by the hardware when an unexpected write to the AES_DINR register is detected (during the evaluation or data output phase). While AES is running, try changing CHMOD, MODE,This bit is also set for DATATYPE, AES_KEY *, or AES_IVR *. Even if WERR is set, this flag has no effect on the AES that continues to run. Even with WERR set, the flag has no effect on AES that continue to run. It is cleared by the software by setting the ERRC bit in the AES_CR register. 0: No write error was detected 1: A write error is detected

[1]	RW	**RDERR:** Read error flag This bit is set by the hardware when an unexpected read operation from the AES_DOUTR register is detected (during the calculation or data entry phase). If the ERRIE bit was previously set in the AES_CR register, an interrupt is generated. This flag is used for AES that will continue to run even if RDERR is set. It is cleared by the software by setting the ERRC bit in the AES_CR register. 0: No read error was detected 1: A read error is detected.
[1]	RW	**CCF:** Calculation completion mark This bit is set by the hardware when the calculation is complete, and an interrupt is generated if the CCFIE bit was previously set in the AES_CR register. It is cleared by the software by setting the CCFC bit in the AES_CR register. 0: The calculation is complete 1: The calculation is incomplete **Note:** *This bit is only valid if DMAOUTEN = 0. It may remain high when DMAOUTEN = 1.*

29.9.3 AES data input register(AES_DINR)

Register	Address offset	Access	Reset value	Description
AES_DINR	0x08	R	0x0000_0000	AES data input register

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

AES data input register(AES_DINR)bit description

Bit	Access	Description
[31:0]	R	**DINR[31:0]:** Data input register. This register must be written four times in the input phase: In mode 1 (encryption), four words must be written, representing plain text from MSB to LSB. In mode 2 (key derivation), this register is not used because this mode only involves derived key computation starting from the AES_KEYRx register. In mode 3 (decryption) and Mode 4 (Key derivation + decryption), four words must be written, representing ciphertext MSB to LSB.

29.9.4 AES data output register(AES_OUTR)

Register	Address offset	Access	Reset value	Description
AES_OUTR	0x0C	R	0x0000_0000	AES data output register

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

AES data output register(AES_OUTR)bit description

Bit	Access	Description
[31:0]	R	**DOUTR[31:0]:** Data output register The register is read-only. Once the CCF flag is set, the 128-bit output can be accessed by reading the data register four times: In mode 1 (encryption), the four words read represent the ciphertext from MSB to LSB. In mode 2 (key derivation), there is no need to read this register because the derived key is located in the AES_KEYRx register. In mode 3 (decryption) and Mode 4 (key derivation + decryption), the four words read represent plain text from MSB to LSB

29.9.5 AES key register0(AES_KEYR0)(key[31:0])

Register	Address offset	Access	Reset value	Description
AES_KEYR0	0x10	RW	0x0000_0000	AES key register0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR0[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR0[15:0]															

AES key register0(AES_KEYR0)(key[31:0])bit description

Bit	Access	Description
[31:0]	RW	**KEYR0[31:0]:** Key register (LSB key [31:0])

29.9.6 AES key register1(AES_KEYR1)(LSB: key[63:32])

Register	Address offset	Access	Reset value	Description
AES_KEYR1	0x14	RW	0x0000_0000	AES key register1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR1[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR1[15:0]															

AES key register1(AES_KEYR1)(LSB: key[63:32])bit description

Bit	Access	Description
[31:0]	RW	**KEYR1[31:0]:** AES Key Register (Key [63:32])

29.9.7 AES key register2(AES_KEYR2)(key[95:64])

Register	Address offset	Access	Reset value	Description
AES_KEYR2	0x18	RW	0x0000_0000	AES key register2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR2[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR2[15:0]															

AES key register2(AES_KEYR2)(key[95:64])bit description

Bit	Access	Description
[31:0]	RW	**KEYR2[31:0]:** AES Key Register (Key [95:64])

29.9.8 AES key register3(AES_KEYR3)(key[127:96])

Register	Address offset	Access	Reset value	Description
AES_KEYR3	0x1C	RW	0x0000_0000	AES key register3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR3[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYR3[15:0]															

AES key register3(AES_KEYR3)(key[127:96])bit description

Bit	Access	Description
[31:0]	RW	**KEYR3[31:0]:** AES Key Register (Key [127:96])

29.9.9 AES Initializes the vector register0(AES_IVR0)(IVR[31:0])

Register	Address offset	Access	Reset value	Description
AES_IVR0	0x20	RW	0x0000_0000	AES Initializes the vector register0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR0[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVR0[15:0]															

AES Initializes the vector register0(AES_IVR0)(IVR[31:0])bit description

Bit	Access	Description
[31:0]	RW	**IVR0[31:0]:** Initialize the vector register (LSB IVR[31:0]) This register must be written before the EN bit in the AES_CR register is set

29.9.10 AES Initializes the vector register1(AES_IVR1)(IVR[63:32])

Register	Address offset	Access	Reset value	Description
AES_IVR1	0x24	RW	0x0000_0000	AES Initializes the vector register1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR1[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVR1[15:0]															

AES Initializes the vector register1(AES_IVR1)(IVR[63:32])bit description

Bit	Access	Description
[31:0]	RW	**IVR1[31:0]:** Initialize the vector register (IVR[63:32]) This register must be written before the EN bit in the AES_CR register is set

29.9.11 AES Initializes the vector register2(AES_IVR2)(IVR[95:64])

Register	Address offset	Access	Reset value	Description
AES_IVR2	0x28	RW	0x0000_0000	AES Initializes the vector register2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR2[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVR2[15:0]															

AES Initializes the vector register2(AES_IVR2)(IVR[95:64])bit description

Bit	Access	Description
[31:0]	RW	**IVR2[31:0]:** Initialize the vector register(IVR[95:64]) This register must be written before the EN bit in the AES_CR register is set

29.9.12 AES Initializes the vector register3(AES_IVR3)(IVR[127:96])

Register	Address offset	Access	Reset value	Description
AES_IVR3	0x2C	RW	0x0000_0000	AES Initializes the vector register3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR3[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVR3[15:0]															

AES Initializes the vector register3(AES_IVR3)(IVR[127:96])bit description

Bit	Access	Description
[31:0]	RW	**IVR3[31:0]:** Initialize the vector register(MSB IVR[127:96]) This register must be written before the EN bit in the AES_CR register is set

30 Debug support (DBG)

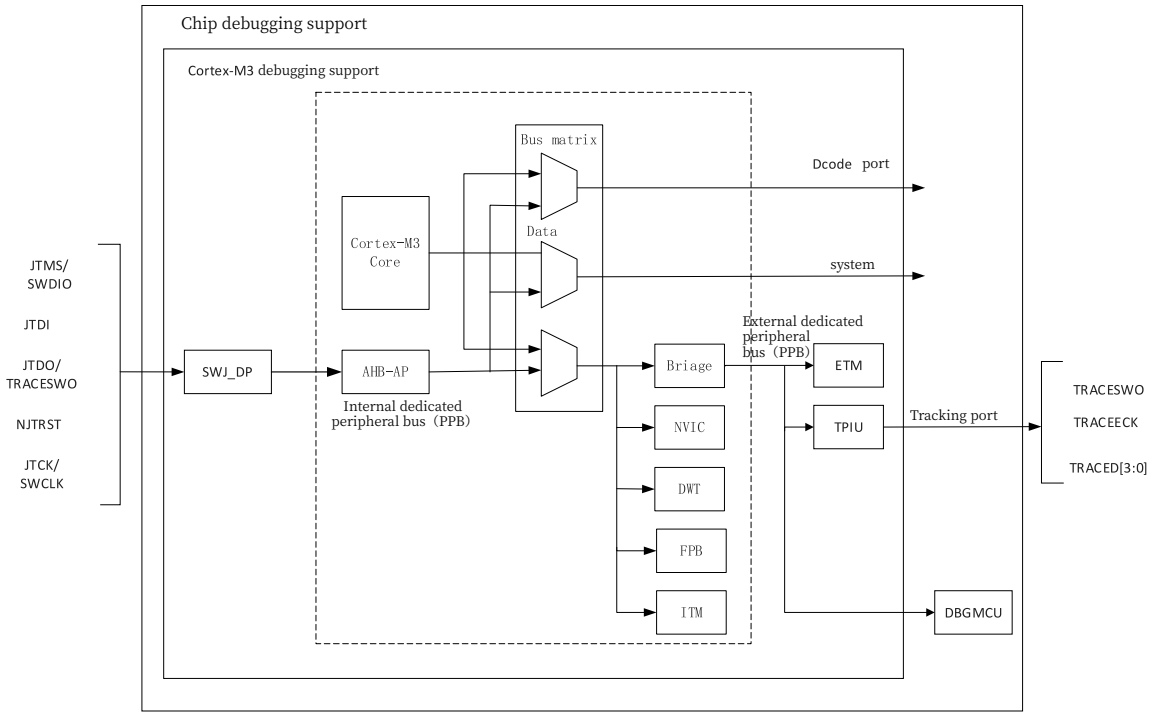
30.1 DBG Introduction

The chip uses the Cortex™-M3 kernel, which contains a hardware debugging module to support complex debugging operations. The hardware debugging module allows the kernel to stop when taking a finger (instruction breakpoint) or accessing data (data breakpoint). When the kernel is stopped, both the internal state of the kernel and the external state of the system can be queried. After the query is complete, the kernel and peripherals can be restored and the program continues.

When the chip connects to the debugger and starts debugging, the debugger uses the kernel's hardware debugging module to perform debugging operations.

The chip supports the Serial debugging interface (SW, Serial Wire).

Fig 30.1-1 Debugging block diagram



30.2 SW debug port

The chip supports only SW debugging, not JTAG debugging.
The following table shows the pins of SW-DP (Serial Wire Debug Port)

SW debugging interface			Pin assignment
Name	Type	Debugging function	
SWDIO	I/O	Serial data input/output	PA13
SWCLK	I	Serial clock	PA14

After the system is reset, SWDIO and SWCLK are initialized as dedicated pins that can be used by the debugger (note that trace output pins are not initialized unless defined by the debugger). You can disable the pin functionality of the SW-DP interface by reusing the remapping and debugging I/O configuration register (AFIO_MAPR) registers. These dedicated pins will be released for use as ordinary I/O ports. This register is mapped to the APB bridge connected to the Cortex™-M3 system bus.

Setting this register is done by user code, not by the debugger.

Three control bits are used to configure the pins of the SWJ-DP interface. These three control bits are reset when the system is reset.

AFIO_MAPR(Address: 0x40010004)

- Read: APB, no wait state
- Write: APB, a wait state if the AHB-APB bridge write buffer is full

Bit 26:24 = SWJ_CFG[2:0] is set and reset by the software. The reset value is 0x4. By default, PA13 and PA14 are SWD functions.

30.3 ID code and locking mechanism

The chip contains an MCU ID code, which is a component of DBG_MCU and is mapped to the PPB bus, which can be accessed with the SW debug port or through user code. This code can also be accessed when the MCU is in the system reset state.
Register name: DBGMCU_IDCODE
Address: 0xE004_2000
Only 32-bit access is supported. The read-only value is 0x10000414 and cannot be written.

30.4 Trace Port Interface Uint

30.4.1 Asynchronous mode

Asynchronous mode requires an additional pin and is present in all packages.

Tab 30.4-1 Asynchronous trace pin assignment

Pin name	Reserved		Pin assignment
	Type	Description	
TRACESWO	output	Asynchronous trace data output	PB3

30.4.2 Synchronous mode

The synchronous mode uses five additional pins depending on the length of the trace data and is only available in a 100-pin package

Tab 30.4-2 Synchronous tracking pin assignment

Pin name	Reserved		Pin assignment
	Type	Description	
TRACECK	output	Asynchronous trace data output	PE2
TRACED[3:0]	output	Synchronous tracking data output,The length can be 1, 2, or 4	PE2

These pins are not dedicated pins by default and are assigned by setting the TRACE_IOEN and TRACE_MODE bits of the MCUDBG register

30.5 MCU debugging module (MCUDBG)

The MCU debug module assists the debugger with the following functions:

- Low power consumption mode
- Provides clock control with timer, watchdog, I2C and CAN at the breaking point
- Control the distribution of the trailing foot

30.5.1 Debugging support in low power mode

Low power mode can be entered using WFI and WFE.
The MCU supports a variety of low-power modes that can turn off the CPU clock or reduce the CPU's power consumption.
The debugger can change the characteristics of low power mode by configuring registers.
In sleep mode, the debugger must first set the DBG_SLEEP bit of the DBGMCU_CR register. This will provide HCLK with the same clock as FCLK(the system clock configured by code).

30.5.2 Supports debugging of timer, watchdog, CAN, and I2C

When a breakpoint is generated, it is necessary to select the operating mode of the counter according to the different uses of the timer and the watchdog:

- When a breakpoint is generated, the counter continues counting. This is often used when output PWM control motor. When a breakpoint is generated, the counter stops counting. This is required for watchdog counters.
- With CAN, the user has the option to block updates to the receive register during the breakpoint.
- For I2C, the user can choose to prevent SMBUS timeouts during breakpoints

30.5.3 Debug the MCU control register(DBGMCU_CR)

Register	Address offset	Access	Reset value
DBGMCU_CR	0xE004_2004	RW	0x0000_0000

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
Reserved											DBG_TIMx_STOP			DBG_TIMx_STOP	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	DBG_CAN_STOP	DBG_TIMx_STOP				DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE[1:0]		TRACE_IOEN	Reserved		Set to 0	Set to 0	DBG_SLEEP

Debug the MCU control register(DBGMCU_CR)bit description

Bit	Access	Description
[31:21]	R	Reserved
[20:18]	RW	**DBG_TIMx_STOP:** Stop timer counter when kernel stops (x=7.. 5) 0: When the kernel stops, the clock is still supplied to the counter of the associated timer, and the timer output works properly 1: When the kernel stops, turns off the clock on the counter of the associated timer and turns off the output of the timer (as if in an emergency response to a pause event, stopping the timer).
[17]	RW	**DBG_TIMx_STOP:** Stop timer counter when kernel stops (x=8) 0: When the kernel stops, the clock is still supplied to the counter of the associated timer, and the timer output works properly 1: When the kernel stops, turns off the clock on the counter of the associated timer and turns off the output of the timer (as if in an emergency response to a pause event, stopping the timer).
[16]	RW	**DBG_I2C2_SMBUS_TIMEOUT:** Stop SMBUS timeout mode when the kernel stops. 0: The operation is the same as the normal mode. 1: Freezes the timeout control of the SMBUS
[15]	RW	**DBG_I2C1_SMBUS_TIMEOUT:** Stop SMBUS timeout mode when the kernel stops. 0: The operation is the same as the normal mode. 1: Freezes the timeout control of the SMBUS
[14]	RW	**DBG_CAN_STOP:** CAN stops running when the kernel is in debug state. 0: CAN still runs normally. 1: The receive register of CAN does not continue to receive data.
[13:10]	RW	**DBG_TIMx_STOP:** Counter stops working when kernel enters debug state x=4.. 1. 0: indicates that the timer counter is still working properly. 1: Select the timer counter to stop working.
[9]	RW	**DBG_WWDG_STOP:** The debug window watchdog stops working when the kernel is in debug state. 0: The window watchdog counter is still working properly. 1: The window watchdog counter stops working.

[8]	RW	**DBG_IWDG_STOP:** The independent watchdog stops working when the kernel enters the debug state 0: The independent watchdog counter is still working properly. 1: The independent watchdog counter stops working.
[7:5]	RW	**DBG_IWDG_STOP[1:0],TRACE_IOEN:** Tracking pin allocation control When TRACE_IOEN = 0: TRACE_MODE = xx: Tracing pins are not assigned (default state). When TRACE_IOEN = 1: TRACE_MODE = 00: The tracing pin uses asynchronous mode; (SWJ_CFG[2]=0 is required.) TRACE_MODE=01: The tracking pin uses synchronous mode and has a data length of 1; TRACE_MODE=10: The tracking pin uses synchronous mode and has a data length of 2; TRACE_MODE=11: The tracing pin uses synchronous mode and has a data length of 4.
[4:3]	R	Reserved
[2]	RW	For internal use, the value must be set to 0
[1]	RW	For internal use, the value must be set to 0
[0]	RW	**DBG_SLEEP:** Debug sleep mode 0: (FCLK on, HCLK off) In sleep mode, FCLK is provided by the previously 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 system when exiting from sleep mode. 1: (FCLK on, HCLK on) In sleep mode, the FCLK and HCLK clocks are provided by the previously configured system clock.

31 Version history

Revision	Data	Changes
01.00	2022/12/15	Draft version
02.00	2022/12/18	Update clock tree
02.10	2023/7/13	Frequency splitting Settings for different baud rates
02.20	2023/8/25	Update cmd register
02.30	2023/10/25	Update DMA_CFGHx register
02.40	2023/10/27	Update DMA register
02.50	2026/01/15	Update Chapter 22.3.5 and add Note2

IMPORTANT EXPLANATION

Please Read Carefully:

Information in this document is provided solely in connection with MG products.This document, including anyproduct of MG described in this document (the ”Product”), is owned by MG under the intellectualproperty laws and treaties of the People’s Republic of China and other jurisdictions worldwide. Megawin and its subsidiaries(”MG”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.MG does not assume any liability arising out of the application or use of any Product described in this document.

Purchasers are solely responsible for the choice, selection and use of the MG products and services described herein, and MG assumes no liability whatsoever relating to the choice, selection or use of the MG products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by MG for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed or manufactured for ordinary business, industrial, personal, or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage.

Resale of MG products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by MG for the MG product or service described herein and shall not create or extend in any manner whatsoever, any liability of MG.

©2026 Megawin All Rights Reserved.