TPS320A

Technical Reference Manual

TRM320001A00E
Nov. 2023

List of Figures

List of Tables

Read This First

About This Manual

This technical reference manual provides detailed information including how to use the TPS325M51xx device, system and bus architecture, memory organization and peripheral instructions. The target audiences for this document are software developers, application developers and hardware developers.

For more information regarding pin assignment, package and electrical characteristics, refer to the datasheet.

Glossary

ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
ADVTMR	Advanced-Control Timer
BPU	Breakpoint Unit
BOR	Brownout Reset
CBC	Cipher Block Chaining
CAN	Controller Area Network
CFB	Cipher Feedback
CMP	Comparator
CRC	Cyclic Redundancy Check
DAC	Digital-to-Analog Converter
DMA	Direct Memory Access
DWT	Data Watchpoint and Trace
ECB	Electronic Codebook
EXTI	Extended Interrupts and Events Controller
ETM	Embedded Trace Macrocell
EHS	External High Speed
FIPS	Federal Information Processing Standard
FIFO	First-In First-Out
FMC	Flash Memory Controller
GPIO	General-Purpose I/Os
GPTMR	General-Purpose Timer
IWDG	Independent Watchdog

I2S	Integrated Interchip Sound
I2C	Inter-Integrated Circuit
ITM	Instrumentation Trace Macrocell
LFSR	Linear Feedback Shift Register
LIN	Local Interconnection Network
NVIC	Nested Vectored Interrupt Controller
OA	Operational Amplifier
OFB	Output Feedback
PCROP	Proprietary Code Readout Protection
PMU	Power Management Unit
POR	Power-On Reset
PDR	Power-Down Reset
QSPI	Quad Serial Peripheral Interface
RDP	Read Protection
RWW	Read-While-Write
RTC	Real-Time Clock
RCC	Reset and Clock Control
SPI	Serial Peripheral Interface
SMBus	System Management Bus
TPIU	Trace Port Interface Unit
TPA	Trace Port Analyzer
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
VREFBUF	Voltage Reference Buffer
WWDG	Window Watchdog
WRP	Write Protection

List of Abbreviations for Registers

Read/Write (RW)	Software can read and write to this bit.
Write/Read (WR)	Software can read and write to this bit.
Read-only (RO)	Software can only read this bit.
Write Clear (WC)	Software can clear this bit by writing to the register.
Read/Write Clear (RC_W1)	Software can read and clear this bit by writing '1' to the register.

Reserved Reserved bit, must be kept at reset value.

Conventions: Notes and Cautions

NOTE: Emphasizes information that deserves extra attention.

CAUTION

This operation could lead to severe undesirable outcomes, such as damage to the chip or irreversible malfunction.

SOC Overview

This chapter describes the details of the SOC top level overview.

Topics: **Page**

1.1 Introduction

The SOC overview introduces MCU core, bus matrix, memory map, non-volatile memory, SRAM, boot mode, and bootloader.

1.2 MCU Core

The STAR processor is a high-performance 32-bit processor designed for the microcontroller market. The processor offers outstanding performance, fast interrupt handling, and enhanced system debugging with extensive breakpoint and trace capabilities.

The STAR processor is built on a high-performance processor core with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The in-order processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design.

The STAR processor provides high-end processing hardware including:

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

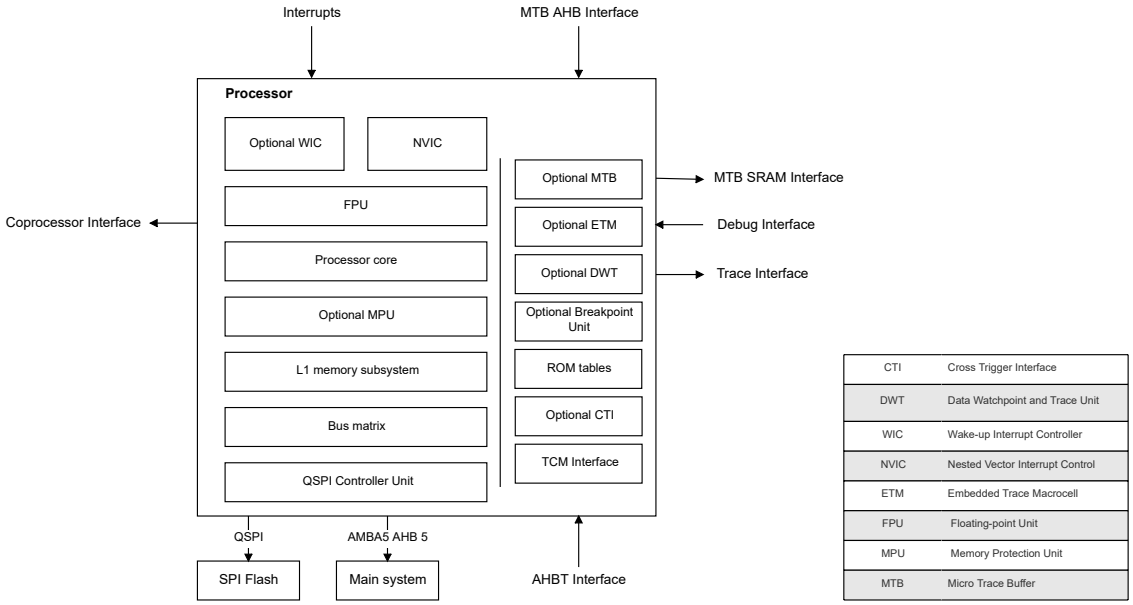


Figure 1-1 Structure of ARM STAR Core Processor

1.3 System Architecture

The main system comprises interconnected 32-bit multi-layer AHB buses:

- Six masters:
  - Arm STAR core C-AHB
  - Arm STAR core S-AHB
  - DMA0 AHB master1 (memory and peripheral)
  - DMA0 AHB master2 (memory and peripheral)
  - DMA1 AHB master1 (memory and peripheral)
  - DMA1 AHB master2 (memory and peripheral)
- Six slaves:
  - Internal Flash memory
  - Internal SRAM0
  - Internal SRAM1
  - Internal SRAM2
  - AHB1 peripherals including AHB to APB bridges and APB peripherals
  - AHB2 peripherals

The bus matrix provides master-slave access to support parallel access and efficient operation, even when multiple high-speed peripherals operate simultaneously.

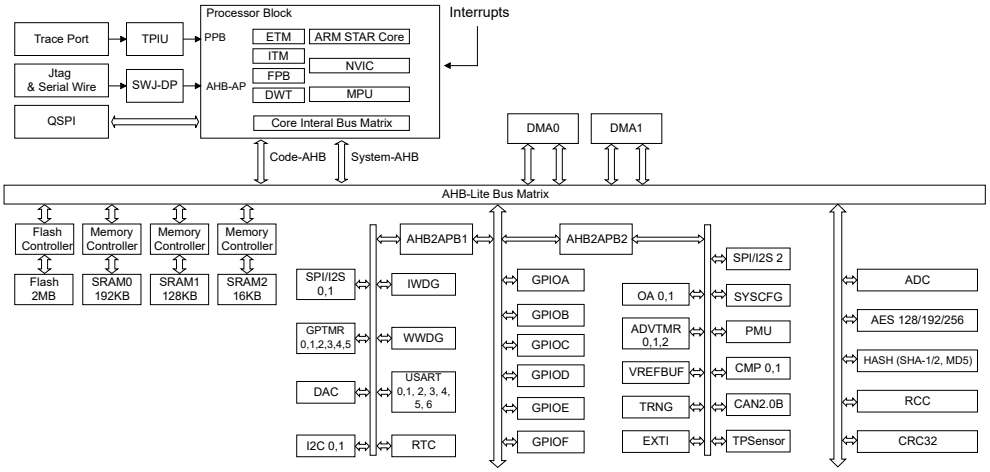


Figure 1-2 System Block Diagram

1.3.1 Code AHB Interface

The Code AHB (C-AHB) interface is used for any instruction fetch and data access to the code region of the memory map.

1.3.2 System AHB Interface

The System AHB (S-AHB) interface is used for data access to the SRAM, peripheral and external device, or system regions of the memory map.

1.3.3 DMA Bus

The DMA bus connects the AHB master interface of the DMA to the bus matrix. The bus targets Flash memory, SRAM, and AHB1 peripherals, including APB1 and APB2 peripherals, as well as AHB2 peripherals.

1.3.4 Bus Matrix

The bus matrix consists of six masters: C-AHB, S-AHB, DMA0, and DMA1 master1/2. It also includes six slaves: Flash, SRAM0/1/2, and peripherals. The bus matrix is responsible for coordinating access arbitration between the CPU and DMA buses using a Round-robin algorithm.

1.3.5 AHB/APB Bridges

Two AHB/APB Bridges provide a fully synchronous connection between the AHB and the two APB buses, allowing flexible selection of peripheral frequency up to 156 MHz. After the device is reset, all peripheral clocks, except for SRAM and Flash, are disabled. Before using a peripheral, the user has to enable its clock through the Reset and Clock Control (RCC).

1.4 Memory Organization

1.4.1 Introduction

The Main Flash, system memory, SRAM, and peripheral registers are organized in a linear 4 GB address space. The memory space is divided into eight main blocks, with each block being 512 MB in size.

Bytes within the memory are organized in little-endian format. This means that the lowest numbered byte in a word is considered the least significant byte, while the highest numbered byte is considered the most significant byte.

1.4.2 Memory Map and Register Boundary Addresses

Figure 1-3 shows the memory map.

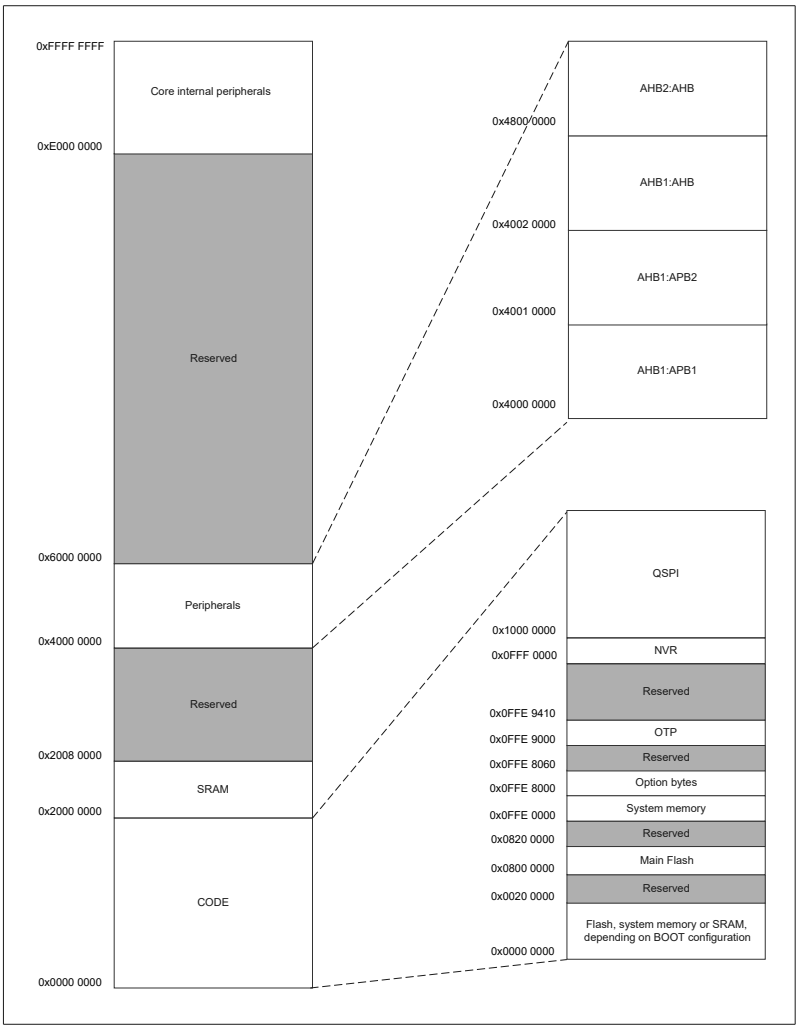


Figure 1-3 Memory Map

All memory map areas that are not allocated to on-chip memories and peripherals are considered Reserved. For a detailed mapping of the available memory and register areas, please refer to the device datasheet.

1.5 Flash Memory

Flash memory is composed of three distinct physical areas:

- The main Flash area, which contains the application program and user data.
- The NVR area, which also contains the application program and user data.

NOTE: Switching access between the main Flash and NVR region may result in performance degradation.

- The information area, which is further divided into three parts:
  - Option bytes for hardware and memory protection user configuration.
  - System memory that contains 3PEAK proprietary code, such as bootloader.
  - OTP (One-Time Programmable).

The Flash interface implements instruction access and data access based on the AHB protocol. It also implements the logic needed to perform Flash memory operations (erase/program) controlled through the FMC registers.

1.6 Embedded SRAM

The TPS325M51xx device has up to 336 Kbytes of SRAM, which is divided into the following sections:

- 192 Kbytes of SRAM0, mapped at address 0x2000 0000
- 128 Kbytes of SRAM1, mapped at address 0x2003 0000
- 16 Kbytes of retention SRAM2, mapped at address 0x2005 0000

The SRAM can be accessed by the CPU and DMA using byte, half-word, or full word at the maximum system clock frequency, without any wait.

To enhance execution performance, the CPU can access SRAM0 through the ITCM/DTCM buses when TCM is enabled.

1.7 Boot Configuration

1.7.1 Boot Mode

Three different boot modes are shown below:

- Boot from main Flash: the main Flash memory is aliased in the boot memory space (0x0000 0000) and is still accessible from its original memory space (0x0800 0000 bank0).
- Boot from system Flash: the system memory is aliased in the boot memory space (0x0000 0000) and is still accessible from its original memory space (0x0FFE 0000).
- Boot from the embedded SRAM0: the SRAM0 is aliased in the boot memory space (0x0000 0000) and is still accessible from its original memory space (0x2000 0000).

After the boot delay, the CPU fetches the top-of-stack value from address 0x0000 0000 and executes code from the boot memory at 0x0000 0004.

NOTE: When the device boots from SRAM, in the application initialization code, it is necessary to relocate the vector table in SRAM using the NVIC exception table and the offset register.

Once the boot mode is selected, the application software can modify the option byte and then execute OBLaunch or power on reset to change the boot mode. This modification is performed by programming the FMC_OPTR register.

1.7.2 Physical Remap

The following memories can be remapped:

- Main Flash memory
- System memory
- Embedded SRAM0

Table 1-1 Memory Mapping vs. Boot Mode/Physical Remap

Addresses	Boot/Remap in Main Flash Memory	Boot/Remap in Embedded SRAM0	Boot/Remap in System Memory
0x20000000 - 0x2002 FFFF	SRAM0	SRAM0	SRAM0

Addresses	Boot/Remap in Main Flash Memory	Boot/Remap in Embedded SRAM0	Boot/Remap in System Memory
0x0FFE 0000 - 0x0FFE 9FFF	System memory/OTP/ options bytes	System memory/OTP/ options bytes	System memory/OTP/ options bytes
0x0800 0000 - 0x081F FFFF	Flash memory	Flash memory	Flash memory
0x0000 0000 - 0x001F FFFF	Flash (2 MB) aliased	SRAM0 (192KB) Aliased	System memory (64 KB) aliased

1.7.3 Boot Configurations

Those boot modes can be configured through the BOOTLOCK bit or BOOT pin (NBOOT0 bit), NBOOT1 bit, DBANK bit, and SWAPBANK bit in the FMC_OPTR register. The boot pin and boot bits are latched on the 4th edge of the internal startup clock source after the reset release. It is up to the user to set these bits to select the required boot mode.

The BOOTLOCK bit, BOOT pin or NBOOT0 bit (if the NSWBOOT0 bit in the FMC_OPTR register is cleared), NBOOT1 bit, DBANK bit, and SWAPBANK bit are also resampled when exiting from standby mode. Therefore, they must be kept in the required boot mode configuration in standby mode.

The BOOT pin is configured in input mode in the following scenarios:

- During the complete reset phase, if the option bit NSWBOOT0 is set, and then switches automatically in analog mode after the reset is released (BOOT pin).
- From the reset phase to the completion of the option byte loading if the bit NSWBOOT0 is cleared (BOOT0 value coming from NBOOT0 bit). It then automatically switches to the analog mode, even if the reset phase isn't complete.

Table 1-2 Boot Modes Configuration

BOOT LOCK FMC_BOOTR[16]	NSWBOOT0 FMC_OPTR[26]	BOOT Pin	NBOOT0 FMC_OPTR[27]	NBOOT1 FMC_OPTR[23]	DBANK FMC_OPTR[22]	SWAPBANK FMC_OPTR[20]	Boot Memory Space Alias
1	X	X	X	X	X	X	Main Flash memory Bank0 is selected
0	1	0	X	X	0	X	Main Flash memory Bank0 is selected
					1	0	Main Flash memory Bank0 is selected
					1	1	Main Flash memory Bank1 is selected
0	0	X	1	X	0	X	Main Flash memory Bank0 is selected
					1	0	Main Flash memory Bank0 is selected

BOOT LOCK FMC_BOOTR[16]	NSWBOOT0 FMC_OPTR[26]	BOOT Pin	NBOOT0 FMC_OPTR[27]	NBOOT1 FMC_OPTR[23]	DBANK FMC_OPTR[22]	SWAPBANK FMC_OPTR[20]	Boot Memory Space Alias
					1	1	Main Flash memory Bank1 is selected
0	1	1	X	0	X	X	Embedded SRAM0 is selected
0	0	X	0	0	X	X	Embedded SRAM0 is selected
0	1	1	X	1	X	X	System memory is selected
0	0	X	0	1	X	X	System memory is selected

When booting from the main Flash memory, the application software can boot from either bank 0 or bank 1, depending on the setting of the DBANK and SWAPBANK bits. By default, booting from bank 0 is selected.

1.8 Embedded Bootloader

The embedded bootloader is located in the system memory and is programmed by 3PEAK during production. When the device boots from the system memory, the bootloader will jump to execute the user application programmed in the main Flash memory.

System Control Module (SYS)

This chapter describes the details of the System Control Module (SYS).

Topics: **Page**

2.1 Introduction

The System Control Module (SYS) module is available for all devices. It is responsible for the interaction between various modules throughout the system.

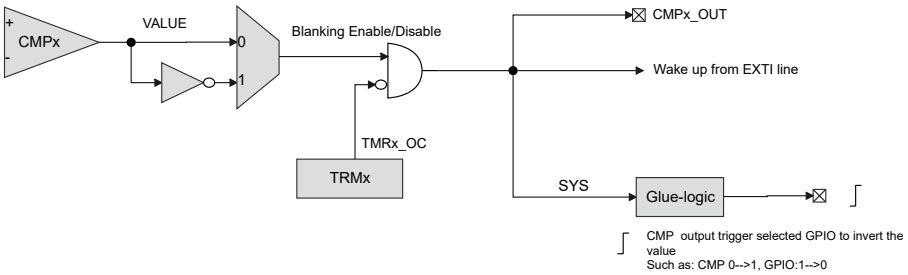
2.2 Features

- On-chip module-to-module signaling control
- Remapping memory areas
- EXTI connection to the GPIOs mapping
- FPU interrupts configuration
- TCM control

2.3 Functional Description

2.3.1 On-chip Module-to-Module Signaling Control

The SOC top-level has the feature to control module-to-module interaction. TPS325Mxx device can route CMP output to a dedicated GPIO.



The polarity from CMP can be set, and GPIO is also configurable. Refer to SYSCFG Registers for more information.

2.3.2 Memory Remap

The chip boot can be controlled by various boot options, such as the external BOOT pin or the registers NBOOT0, NBOOT1, and NSWBOOT0. These options determine whether the chip boots from SRAM, main Flash, or system Flash. For more details, refer to Boot Configurations.

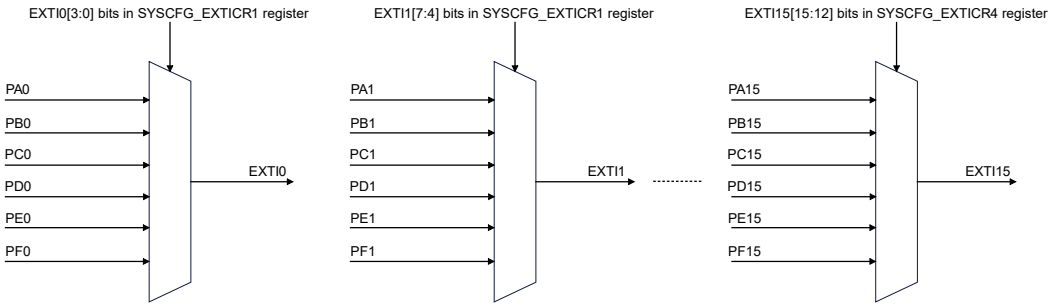
BOOTLOCK	NBOOT1	NBOOT0	BOOT (Pin)	NSWBOOT0	Boot
1	x	x	x	x	Flash
0	x	x	0	1	Flash
	x	1	x	0	Flash
	0	x	1	1	SRAM
	0	0	x	0	SRAM
	1	x	1	1	System Memory (Bootloader)
	1	0	X	0	System Memory (Bootloader)

When the boot configuration is set, the related physical address is automatically remapped.

Address	Boot from Flash	Boot from SRAM	Boot from System Memory
0x2000 0000 - 0x2000 FFFF	SRAM	SRAM	SRAM
0x0FFE 0000 - 0x0FFE 9FFF	System memory	System memory	System memory
0x0800 0000 - 0x081F FFFF	Flash	Flash	Flash
Remapped address depending on boot option			
0x0000 0000 - 0x001F FFFF	Flash	SRAM	System memory

2.3.3 EXTI Connection to GPIO

There is a total of up to 16 configurable EXTI remapping to all possible GPIO, as shown in the block diagram below.



The remapping function is controlled by EXTIx; refer to Registers for more information.

Table 2-1 EXTI Lines Connections

EXTI Line	Line Source	Line Type
0	EXTI0	Configurable
1	EXTI1	Configurable
2	EXTI2	Configurable
3	EXTI3	Configurable
4	EXTI4	Configurable
5	EXTI5	Configurable
6	EXTI6	Configurable
7	EXTI7	Configurable
8	EXTI8	Configurable
9	EXTI9	Configurable
10	EXTI10	Configurable
11	EXTI11	Configurable
12	EXTI12	Configurable

EXTI Line	Line Source	Line Type
13	EXTI13	Configurable
14	EXTI14	Configurable
15	EXTI15	Configurable

2.3.4 FPU Interrupt Configuration

There is one interrupt port for the FPU on the specified device (refer to the device specification for details). However, the FPU has up to 6 different types of interrupts, and the SYS module will provide the enable/disable for each individual interrupt.

FPUIE	FPUSTATUS	FPU Interrupt
FPU_IE[5]: Inexact interrupt enable	FPIOC	FPU Interrupt service routine
FPU_IE[4]: Input denormal interrupt enable	FPDZC	
FPU_IE[3]: Overflow interrupt enable	FPUFC	
FPU_IE[2]: Underflow interrupt enable	FPOFC	
FPU_IE[1]: Divide-by-zero interrupt enable	FPIDC	
FPU_IE[0]: Invalid operation interrupt enable	FPIXC	

2.3.5 TCM Control

The TCM function can be enabled or disabled via the TCMEB register. To optimize the use of system SRAM, TCM is sharing with system SRAM, starting from the address 0x2000 0000. The specified addresses are as follows when DTCMEB = 1 and ITCMEB = 1.

- ITCM: 0x2000 0000-0x2000 FFFF
- DTCM: 0x2001 0000-0x2001 FFFF

In terms of TCM structure, the size of DTCM is half of ITCM. When ITCM is enabled by setting ITCMEB to 1, DTCM is divided into two parts: D0TCM (32 KB) and D1TCM (32 KB).

- When TCM is disabled, the core can access system SRAM through path1.
- When TCM is enabled, TCM can access system SRAM through path2, and the core can fetch the instruction from TCM. In this case, it is not able to directly access the system SRAM on path1.

Figure 2-1 shows the TCM sharing with the system SRAM block diagram.

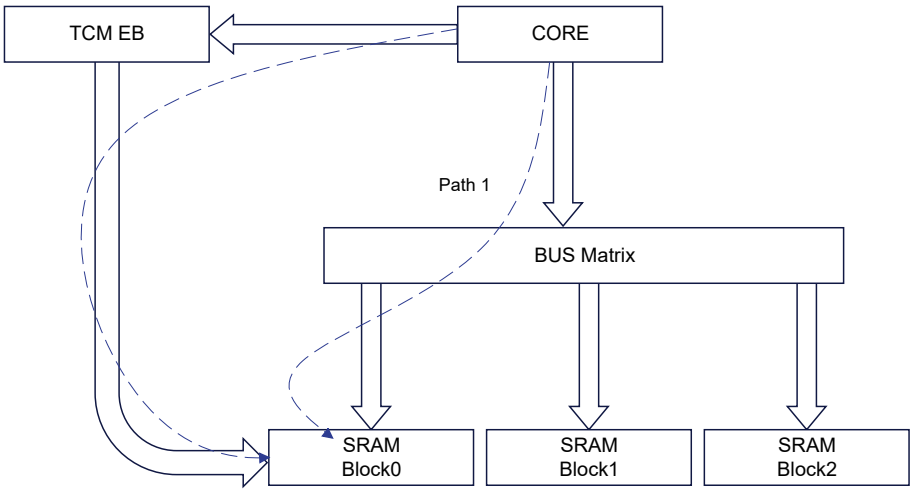


Figure 2-1 TCM Access System SRAM

2.4 Registers

2.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	SYSCFG_MEMRMP	Memory remap register
0x0004	SYSCFG_CFGR1	Configuration register 1
0x0008	SYSCFG_EXTICR1	EXTI GPIO select register 1
0x000c	SYSCFG_EXTICR2	EXTI GPIO select register 2
0x0010	SYSCFG_EXTICR3	EXTI GPIO select register 3
0x0014	SYSCFG_EXTICR4	EXTI GPIO select register 4
0x0024	SYSCFG_INSTCTRL	Instruction execution control register
0x002c	SYSCFG_TRACECTRL	Trace control register
0x0038	SYSCFG_PINOVERTURNINGCTRL0	Pin overtruning control register for CMP0
0x003c	SYSCFG_PINOVERTURNINGCTRL1	Pin overtruning control register for CMP1
0x0044	SYSCFG_TCMEB	TCM enable signal
0x0050	SYSCFG_CANCNTCTRL	CAN timer control register
0x0100	SYSCFG_OPENINFO0	Device ID info register
0x0104	SYSCFG_OPENINFO1	Hardware revision and fimware revision register
0x0108	SYSCFG_OPENINFO2	DIE X position and DIE Y position register
0x010c	SYSCFG_OPENINFO3	LOT wafer ID register

2.4.2 Register Field Details

2.4.2.1 SYSCFG_MEMRMP

0x0000			Memory remap register											SYSCFG_MEMRMP		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							FBMODE	Reserved						MEMMODE	
Type	RO							RO	RO						RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-2 Memory Remap Register Description

Field	Name	Description
31:9	Reserved	Reserved
8	FBMODE	Flash Bank mode selection 0: Flash Bank 1 mapped at 0x0800 0000 (and aliased @0x0000 0000(1)) and Flash Bank 2 mapped at 0x0804 0000 (and aliased at 0x0008 0000) 1: Flash Bank 2 mapped at 0x0800 0000 (and aliased @0x0000 0000(1)) and Flash Bank 1 mapped at 0x0804 0000 (and aliased at 0x0008 0000)
7:2	Reserved	Reserved
1:0	MEMMODE	Memory mapping selection These bits control the memory internal mapping at address 0x0000 0000. These bits are used to select the physical remap by software and bypass the BOOT pin and the option bit

Field	Name	Description
		setting. After resetting, these bits take the value selected by the BOOT0 pin (or option bit nSWBOOT0) and BOOT1 option bit. 0b000: Main Flash memory mapped at 0x00000000 0b001: SRAM1 mapped at 0x00000000 0b010: Reserved 0b011: System Flash memory mapped at 0x00000000

2.4.2.2 SYSCFG_CFGR1

0x0004			Configuration register 1											SYSCFG_CFGR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										FPUIE					
Type	RO										RW					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-3 Configuration Register 1 Description

Field	Name	Description
31:6	Reserved	Reserved
5:0	FPUIE	Floating point unit interrupts enable bits FPUIE[5]: Inexact interrupt enable FPUIE[4]: Input denormal interrupt enable FPUIE[3]: Overflow interrupt enable FPUIE[2]: Underflow interrupt enable FPUIE[1]: Divide-by-zero interrupt enable FPUIE[0]: Invalid operation interrupt enable

2.4.2.3 SYSCFG_EXTICR1

0x0008			EXTI GPIO select register 1											SYSCFG_EXTICR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EXTI3				EXTI2				EXTI1				EXTI0			
Type	RW				RW				RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-4 EXTI GPIO Select Register 1 Description

Field	Name	Description
31:16	Reserved	Reserved
15:12	EXTI3	These bits are written by software to select the source input 0000: PA[3] pin 0001: PB[3] pin 0010: PC[3] pin 0011: PD[3] pin 0100: PE[3] pin 0101: PF[3] pin
11:8	EXTI2	These bits are written by software to select the source input 0000: PA[2] pin 0001: PB[2] pin 0010: PC[2] pin 0011: PD[2] pin

Field	Name	Description
		0100: PE[2] pin 0101: PF[2] pin
7:4	EXTI1	These bits are written by software to select the source input 0000: PA[1] pin 0001: PB[1] pin 0010: PC[1] pin 0011: PD[1] pin 0100: PE[1] pin 0101: PF[1] pin
3:0	EXTI0	These bits are written by software to select the source input 0000: PA[0] pin 0001: PB[0] pin 0010: PC[0] pin 0011: PD[0] pin 0100: PE[0] pin 0101: PF[0] pin

2.4.2.4 SYSCFG_EXTICR2

0x000c			EXTI GPIO select register 2											SYSCFG_EXTICR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EXTI7				EXTI6				EXTI5				EXTI4			
Type	RW				RW				RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-5 EXTI GPIO Select Register 2 Description

Field	Name	Description
31:16	Reserved	Reserved
15:12	EXTI7	These bits are written by software to select the source input 0000: PA[7] pin 0001: PB[7] pin 0010: PC[7] pin 0011: PD[7] pin 0100: PE[7] pin 0101: PF[7] pin
11:8	EXTI6	These bits are written by software to select the source input 0000: PA[6] pin 0001: PB[6] pin 0010: PC[6] pin 0011: PD[6] pin

Field	Name	Description
		0100: PE[6] pin 0101: PF[6] pin
7:4	EXTI5	These bits are written by software to select the source input 0000: PA[5] pin 0001: PB[5] pin 0010: PC[5] pin 0011: PD[5] pin 0100: PE[5] pin 0101: PF[5] pin
3:0	EXTI4	These bits are written by software to select the source input 0000: PA[4] pin 0001: PB[4] pin 0010: PC[4] pin 0011: PD[4] pin 0100: PE[4] pin 0101: PF[4] pin

2.4.2.5 SYSCFG_EXTICR3

0x0010			EXTI GPIO select register 3											SYSCFG_EXTICR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EXTI11				EXTI10				EXTI9				EXTI8			
Type	RW				RW				RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-6 EXTI GPIO Select Register 3 Description

Field	Name	Description
31:16	Reserved	Reserved
15:12	EXTI11	These bits are written by software to select the source input 0000: PA[11] pin 0001: PB[11 pin 0010: PC[11] pin 0011: PD[11] pin 0100: PE[11] pin 0101: PF[11] pin
11:8	EXTI10	These bits are written by software to select the source input 0000: PA[10] pin 0001: PB[10] pin 0010: PC[10] pin 0011: PD[10] pin

Field	Name	Description
		0100: PE[10] pin 0101: PF[10] pin
7:4	EXTI9	These bits are written by software to select the source input 0000: PA[9] pin 0001: PB[9] pin 0010: PC[9] pin 0011: PD[9] pin 0100: PE[9] pin 0101: PF[9] pin
3:0	EXTI8	These bits are written by software to select the source input 0000: PA[8] pin 0001: PB[8] pin 0010: PC[8] pin 0011: PD[8] pin 0100: PE[8] pin 0101: PF[8] pin

2.4.2.6 SYSCFG_EXTICR4

0x0014			EXTI GPIO select register 4											SYSCFG_EXTICR4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EXTI15				EXTI14				EXTI13				EXTI12			
Type	RW				RW				RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-7 EXTI GPIO Select Register 4 Description

Field	Name	Description
31:16	Reserved	Reserved
15:12	EXTI15	These bits are written by software to select the source input 0000: PA[15] pin 0001: PB[15] pin 0010: PC[15] pin 0011: PD[15] pin 0100: PE[15] pin 0101: PF[15] pin
11:8	EXTI14	These bits are written by software to select the source input 0000: PA[14] pin 0001: PB[14] pin 0010: PC[14] pin 0011: PD[14] pin

Field	Name	Description
		0100: PE[14] pin 0101: PF[14] pin
7:4	EXTI13	These bits are written by software to select the source input 0000: PA[13] pin 0001: PB[13] pin 0010: PC[13] pin 0011: PD[13] pin 0100: PE[13] pin 0101: PF[13] pin
3:0	EXTI12	These bits are written by software to select the source input 0000: PA[12] pin 0001: PB[12] pin 0010: PC[12] pin 0011: PD[12] pin 0100: PE[12] pin 0101: PF[12] pin

2.4.2.7 SYSCFG_INSTCTRL

0x0024			Instruction execution control register											SYSCFG_INSTCTRL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															CPUWAIT
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-8 Instruction Execution Control Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	CPUWAIT	Stall the core out of reset. The CPUWAIT signal, when HIGH out of reset, forces the core into a quiescent state. The core boot-up sequence and instruction execution are delayed until this signal is driven LOW. During this time, the processor does not perform any memory accesses. Debugger accesses continue when this signal is HIGH.

2.4.2.8 SYSCFG_TRACECTRL

0x002c			Trace control register											SYSCFG_TRACECTRL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved													DAPEN	SOFTSTCLKEN	SOFTTRACEEN
Type	RO													RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Table 2-9 Trace Control Register Description

Field	Name	Description
31:3	Reserved	Reserved
2	DAPEN	Device debug enable, default enable. If it is no longer needed, it can be turned off to save power. 0: DAP disable 1: DAP enable
1	SOFTSTCLKEN	SysTick enable 0: SysTick disable 1: SysTick enable

2.4.2.9 SYSCFG_PINOVERTURNINGCTRL0

0x0038			Pin overtruning control register for CMP0											SYSCFG_PINOVERTURNINGCTRL0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							SELPOLARCMP0	OVERTURNINGEN	GPIOPINSEL				GPIOSEL		
Type	RO							RW	RW	RW				RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-10 Pin Overtruning Control Register for CMP0 Description

Field	Name	Description
31:9	Reserved	Reserved
8	SELPOLARCMP0	Comparator0 polarity selection 0: High level active 1: Low level active
7	OVERTURNINGEN	Overturning GPIO function enable 0: Disable 1: Enable
6:3	GPIOPINSEL	GPIO pin selection 0000: Pin0 0001: Pin1 0010: Pin2 ...

Field	Name	Description
		1111: Pin15
2:0	GPIOSEL	GPIO port selection 0000: GPIOA 0001: GPIOB 0010: GPIOC 0011: GPIOD 0100: GPIOE 0101: GPIOF

2.4.2.10 SYSCFG_PINOVERTURNINGCTRL1

0x003c			Pin overtruning control register for CMP1											SYSCFG_PINOVERTURNINGCTRL1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							SELPOLARCMP1	OVERTURNINGEN	GPIOPINSEL				GPIOSEL		
Type	RO							RW	RW	RW				RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-11 Pin Overtruning Control Register for CMP1 Description

Field	Name	Description
31:9	Reserved	Reserved
8	SELPOLARCMP1	Comparator1 polarity selection 0: High level active 1: Low level active
7	OVERTURNINGEN	Overturning GPIO function enable 0: Disable 1: Enable
6:3	GPIOPINSEL	GPIO pin selection 0000: Pin0 0001: Pin1 0010: Pin2

Field	Name	Description
		... 1111: Pin15
2:0	GPIOSEL	GPIO port selection 0000: GPIOA 0001: GPIOB 0011: GPIOC 0011: GPIOD 0101: GPIOE 0101: GPIOF

2.4.2.11 SYSCFG_TCMEB

0x0044			TCM enable signal											SYSCFG_TCMEB		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														DTCMEB	ITCMEB
Type	RO														RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-12 TCM Enable Signal Register Description

Field	Name	Description
31:2	Reserved	Reserved
1	DTCMEB	DTCM enable 0: Disable 1: Enable
0	ITCMEB	ITCM enable 0: Disable 1: Enable

2.4.2.12 SYSCFG_CANCNTCTRL

0x0050			CAN timer control register											SYSCFG_CANCNTCTRL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														CNTSET0	CNTEN
Type	RO														RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-13 CAN Timer Control Register Description

Field	Name	Description
31:2	Reserved	Reserved
1	CNTSET0	Reset register of CiA 603 timer in CAN 0: Non-reset 1: Reset
0	CNTEN	Enable register of CiA 603 timer in CAN 0: Disable 1: Enable

2.4.2.13 SYSCFG_OPENINFO0

0x0100			Device ID info register											SYSCFG_OPENINFO0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DEVICEID															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DEVICEID															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-14 Device ID Info Register Description

Field	Name	Description
31:0	DEVICEID	Device ID

2.4.2.14 SYSCFG_OPENINFO1

0x0104			Hardware revision and Fimware revision register											SYSCFG_OPENINFO1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	HWREVISION															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FWREVISION															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-15 Hardware Revision and Fimware Revision Register Description

Field	Name	Description
31:16	HWREVISION	Hardware revision
15:0	FWREVISION	Fimware revision

2.4.2.15 SYSCFG_OPENINFO2

0x0108			DIE X position and DIE Y position register											SYSCFG_OPENINFO2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DIEYPOS															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DIEXPOS															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-16 DIE X Position and DIE Y Position Register Description

Field	Name	Description
31:16	DIEYPOS	DIE Y position
15:0	DIEXPOS	DIE X position

2.4.2.16 SYSCFG_OPENINFO3

0x010c			LOT wafer ID register											SYSCFG_OPENINFO3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	LOTWAFERID															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	LOTWAFERID															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-17 LOT Wafer ID Register Description

Field	Name	Description
31:0	LOTWAFERID	LOTWAFERID

Flash Memory Controller (FMC)

This chapter describes the details of the Flash Memory Controller (FMC).

Topics: **Page**

3.1 Introduction

The Flash Memory Controller (FMC) provides all the necessary functions for the on-chip Flash memory. It also provides sector erase, mass erase, and program operations for Flash memory.

3.2 Features

- Up to 2 MB of on-chip Flash memory for instruction and data
- Support for single or dual bank mode
- Read-While-Write capability (RWW) in dual bank mode
- 0 to 4 waiting states when CPU executes instructions
- Prefetch buffer to speed up read operations
- System IBUS cache with 4K bytes, organized as 512 cache line of 2x32 bits
- System DBUS cache with 4K bytes, organized as 256 cache line of 4x32 bits
- Program operation, sector erase, bank erase, and mass erase (both banks)
- One-Time Program (OTP) block used for user data storage
- Option bytes are uploaded to the option bytes control registers when system is reset
- Supports dual boot mode through the SWAPBANK option bit configuration (only in dual bank mode)
- Flash security protection to prevent illegal code/data access:
  - Read protection activated by option (RDP)
  - Write protection areas
  - Proprietary code read protection areas
- Low-power mode and clock autogating function

3.3 Functional Description

3.3.1 Flash Memory Organization

The Flash memory structure is as follows:

Block	Flash Memory Addresses	Size (Bytes)	Name
Main memory (2 MB, DBANK = 1, dual-bank)	0x0800 0000 - 0x0800 0FFF	4K	Sector 0
	0x0800 1000 - 0x0800 1FFF	4K	Sector 1
	0x0800 2000 - 0x0800 2FFF	4K	Sector 2
	0x0800 3000 - 0x0800 3FFF	4K	Sector 3
	-	-	-
	0x080F F000 - 0x080F FFFF	4K	Sector 255
	0x0810 0000 - 0x0810 0FFF	4K	Sector 0
	0x0810 1000 - 0x0810 1FFF	4K	Sector 1
	0x0810 2000 - 0x0810 2FFF	4K	Sector 2
	0x0810 3000 - 0x0810 3FFF	4K	Sector 3
	-	-	-

Block	Flash Memory Addresses	Size (Bytes)	Name
	0x081F F000 - 0x081F FFFF	4K	Sector 255
Main memory (2 MB, DBANK = 0, single-bank)	0x0800 0000 - 0x0800 1FFF	8K	Sector 0
	0x0800 2000 - 0x0800 3FFF	8K	Sector 1
	0x0800 4000 - 0x0800 5FFF	8K	Sector 2
	0x0800 6000 - 0x0800 7FFF	8K	Sector 3
	-	-	-
	0x081F E000 - 0x081F FFFF	8K	Sector 255
System memory	0x0FFE 0000 - 0x0FFE 7FFF	32K	System memory(1)
Option bytes	0x0FFE 8000 - 0x0FFE 805F	96	Option bytes
OTP	0x0FFE 9000 - 0x0FFE 93FF	1K	OTP area(2)
	0x0FFE 9400 - 0x0FFE 940F	16	OTP LOCK(3)

(1) 3PEAK bootloader.

(2) 1 KB One-Time Programmable (OTP) bytes for user data. The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double word cannot be written anymore.

(3) 16 bytes OTP LOCK, each byte lock 64 bytes in the OTP area.

3.3.2 Read Operation

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

Current Buffer

The current buffer is always enabled. Whenever data is read from the Flash memory, it is obtained and stored in the current buffer. The CPU only requires a 32-bit or 16-bit value for each read operation. In the case of sequential code, the next data can be retrieved from the current buffer without the need to fetch it again from Flash memory.

Prefetch

The prefetch buffer is enabled by setting the PRFTEN bit. The prefetch buffer is only performed on IBUS.

In the case of sequential code, when the CPU executes the current buffer data (128-bit), it requires a minimum of 4 clocks for 32-bit operation and at least 8 clocks for 16-bit operation. In this case, the next double-word address data from the Flash memory is prefetched and stored in the prefetch buffer. This ensures that when the CPU finishes processing the current buffer and needs to execute the next set of data, it results in a prefetch buffer hit.

IBUS Cache

The IBUS cache is enabled by setting the IC bit in the ARM CCR register. The IBUS cache is only used when the IBUS fetches data.

If the IBUS data is in the IBUS cache (IBUS cache hit), the CPU reads data from the IBUS cache without any wait states.

If the IBUS data is not in the IBUS cache (IBUS cache miss) and not in the current buffer or prefetch buffer, the cache line is fetched from Flash memory and copied to the IBUS cache. If all cache lines are filled, the LRU (least recently used) policy is used to replace the cache line.

DBUS Cache

The DBUS cache is enabled by setting the DC bit in the ARM CCR register. The DBUS cache is only used when DBUS fetches data by CPU (not by DMA), and the option byte is not cacheable.

If the DBUS data is in the DBUS cache (DBUS cache hit), the CPU reads data from the DBUS cache without any wait states.

If the DBUS data is not in the DBUS cache (DBUS cache miss) and not in the current buffer, the cache line is fetched from Flash memory and copied to the DBUS cache. If all cache lines are filled, the LRU (least recently used) policy is used to replace the cache line.

Table 3-1 TPS32M5xx Buffer Summary

Bank	Current Buffer	Prefetch	D-cache	I-cache
Single bank	128-bit	2x 128-bit	4x 256x32	2x 512x32
Dual bank	64-bit	2x 64-bit		

3.3.3 Autogating Function

The Flash controller has an autogating function that can be enabled to further reduce system power consumption.

NOTE:
- The write clock auto-gating function is enabled by setting the WCLKMODE bit and disabled by clearing the WCLKMODE bit.

  The read clock auto-gating operation (RCLKMODE) is just the opposite. Refer to FMC_ACR for details.
- When the device switches into a low-power mode (stop, standby mode), the auto-gating function must be enabled first.

3.3.4 Unlock FMC_CR

After a reset, the FMC_CR register cannot be accessed in write mode, and the LOCK bit in the FMC_CR register is reset to 1.

The following sequence is used to unlock this register:

1. Write KEY1 = 0x45670123 in the Flash key register (FMC_KEYR).
2. Write KEY2 = 0xCDEF89AB in the FMC_KEYR.
3. The LOCK bit is reset to 0 by hardware after the two write operations.

The software can lock the register again by setting the LOCK bit to 1. If any incorrect operations are performed on the FMC_KEYR, the LOCK bit will be set to 1, a bus error will be detected, and a Hard Fault interrupt will be generated.

NOTE: The FMC_CR register cannot be written when the BUSY bit in the Flash status register (FMC_SR) is set. Any attempt to write to it with the BUSY bit set causes the AHB bus to stall until the BUSY bit is cleared.

3.3.5 Flash Main Memory Erase Sequences

The Flash main memory erase operation can be performed at the sector level, bank level, or on the whole Flash memory (Mass Erase). Mass Erase does not affect the Information block (system Flash, OTP and option bytes).

Sector Erase

To erase a sector, follow the procedure below:

1. Unlock the FMC_CR register.
2. Check the BUSY bit in the FMC_SR register to confirm that no Flash memory operation is in progress (BUSY = 1). Otherwise, wait until the operation has finished.
3. Check and clear all error programming flags due to previous programming. If not, the PGSERR is set.
4. In dual bank mode (DBANK = 1): Set the SER bit and select the sector to erase (SNB) with the associated bank (BKER) in FMC_CR.

   In single bank mode (DBANK = 0): Set the SER bit and select the sector to erase (SNB). The BKER bit in FMC_CR must be kept clear.
5. Set the start erase bit (STRT) in FMC_CR.
6. Wait until all the operations have finished by checking the value of the BUSY bit to be cleared in the FMC_SR register.

NOTE: The internal high speed oscillator IHS (8 MHz) is enabled automatically when the STRT bit is set and disabled automatically when STRT bit is cleared, except if the IHS is previously enabled with IHSON in the RCC_CR register.

If the sector erase is part of write-protected area (by WRP or PCROP), WRPERR is set, and the sector erase request is aborted.

If the mass erase to erase part of write-protected area (by WRP or PCROP), WRPERR is set and the mass erase request is aborted.

Mass Erase

To perform Mass Erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BUSY bit in the FMC_SR.
2. Check and clear all error programming flags due to previous programming. If not, PGSERR is set.
3. For dual bank mode, set the MER0 or MER1 bit in FMC_CR; set only the MER0 bit for single bank mode.
4. Set the STRT bit in the FMC_CR.
5. Wait for the BUSY bit to be cleared in the FMC_SR.

NOTE: The internal high-speed oscillator IHS (8 MHz) is enabled automatically when the STRT bit is set and disabled automatically when the STRT bit is cleared, except if the IHS is previously enabled with IHSON in the RCC_CR register.

The NVR sector mass erase is also performed in mass erase either for single bank mass erase (DBANK = 0, MER0 = 1) or bank 1 mass erase only (DBANK = 1, MER1 = 1) of dual-bank mode.

If the mass erase is to erase part of the write-protected area (by WRP or PCROP), WRPERR is set, and the mass erase request is aborted.

3.3.6 Flash NVR Memory Erase Sequences

To perform NVR sector erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BUSY bit in the FMC_SR.
2. Check and clear all error programming flags due to previous programming. If not, PGSERR is set.
3. Set the BKER, SER, SNB, and NVR bits, whether single bank or dual bank.
4. Select the sector to erase via SNB bits in the FMC_CR.
5. Set the STRT bit in the FMC_CR.
6. Wait for the BUSY bit to be cleared in the FMC_SR.

NOTE: When autogating mode is activated (WCLKMODE bit is set in the FMC_ACR register), the internal high-speed oscillator IHS with a frequency of 8 MHz is automatically enabled when the STRT bit is set. Conversely, it is disabled automatically when the STRT bit is cleared, unless the IHS has been previously turned on using IHSON in the RCC_CR register. The current RDP setting determines the protection of the NVR area.

3.3.7 Flash Memory Programming Sequences

The FMC provides a 64-bit (dual-bank mode) or 128-bit (single bank mode) at a time, which is used to modify the Flash memory contents. Programming in a previously programmed address is not allowed except if the data to write is full zero, and any attempt sets the PROGERR flag in the Flash status register (FMC_SR). It is only possible to program double word (2 x 32-bit data).

- Any attempt to write a byte or half-word sets the SIZERR flag in the FMC_SR.
- Any attempt to write a double word that is not aligned with a double-word address sets the PGAERR flag in the FMC_SR.

Standard Programming

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

1. Check that no Flash memory operation is ongoing by checking the BUSY bit in the FMC_SR.
2. Check and clear all error programming flags due to previous programming. If not, PGSERR is set.
3. Set the PG bit in the FMC control register (FMC_CR).
4. Perform the data write operation at the desired memory address, inside the main memory block, NVR, or OTP area. Only a double word can be programmed (single bank requires a minimum of 4 words):
   - Write the first word in an address aligned with a double word.
   - Write the second word.
5. Wait until the BUSY bit is cleared in the FMC_SR.
6. Check that the EOP flag (this flag is only valid when EOPIE is set in the FMC_SR, indicating that the programming operation has succeeded) and clear it by software.
7. The PG bit will be auto-cleared when the programming request is done.

NOTE: When the Flash interface has received a good sequence (a double word), programming is automatically launched, and the BUSY bit is set. The internal oscillator IHS (8 MHz) is enabled automatically when the PG bit is set and disabled automatically when the PG bit is cleared, except if the IHS is previously enabled with IHSON in the RCC_CR register.

If the user only needs to program one word, the double word must be completed with the erase value 0xFFFF FFFF to launch the programming automatically.

Fast Programming

This mode allows programming a row and reduces the sector programming time by eliminating the need to verify the Flash locations before programming them. It also avoids the rising and falling time of high voltage for each double word. During fast programming, the CPU clock frequency (HCLK) must be at least 32 MHz.

The main memory can be programmed in fast programming mode.

Table 3-2 TPS32M5xx ROW Summary

Bank	ROW
Single bank (DBANK = 0) (1)	256 double words
Dual-bank (DBANK =1) (2)	128 double words

(1) The address is aligned with double word 2 KB boundary.

(2) The address is aligned with double word 1 KB boundary.

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

1. Perform a mass erase. Check that MERFLAG is set. If not, performing fast programming will cause PGSERR to be set.
2. Check that no Flash memory operation is ongoing by checking the BUSY bit in the FMC_SR.
3. Check and clear all error programming flags due to previous programming. If not, PGSERR is set.
4. Set the FSTPG bit in the FMC_CR:
   - Write a first word in an address aligned with double word.
   - Write the second word.
5. Wait until the BUSY bit is cleared in the FMC_SR.
6. Check that the EOP flag (this flag is only valid when EOPIE is set in the FMC_SR register, which means that the programming operation has succeeded) and clear it by software.
7. The FSTPG bit in the FMC_CR will be auto-cleared when a fast programming request is done.

Set PGIE and FSTPG bits in the FMC_CR to enable interrupt fast mode. In ISR routing, write a first word in an address aligned with double word, then write the following words.

NOTE: Ensure the CPU speed is at least 30 MHz to guarantee successful fast page programming.

Configure/enable the DMA channel (refer to DMA Chapter for details) and FSTPG bit to execute DMA fast mode. After the DMA channel receives the request from Flash, the DMA channel will write a word to Flash and then assert an ACK using the standard DMA handshake protocol. Each word write generates a DMA request until a row (512 or 256 words) is written.

Set the PG and NVR bits in the FMC_CR to enable. Perform the data write operation at the desired memory address inside the NVR area. Only double words can be programmed.

NOTE: If the user starts the application code program operation, they must set the MERFLAG bit in the program and erase status register (FMC_PESR) when the application code program is done (all fast programs are completed during the operation).

Don’t try to use fast programming (FSTPG bit) with read operation on the same physical bank. It should use standard programming (PG bit) in this scenario.

When the Flash interface has received the first double word, programming is automatically launched. The BUSY bit is set when applied for the first double word, and it is cleared

when the last double word has been programmed or in case of error. The internal oscillator IHS (8 MHz) is enabled automatically when the FSTPG bit is set, and disabled automatically when the FSTPG bit is cleared, except if the IHS is previously enabled with IHSON in the RCC_CR register.

The 128 double words must be written successively. The voltage is kept on the Flash for all the programming. The maximum time between two double words write requests is the time programming (around 2 x 5us). If a second double word arrives after this time programming, fast programming is interrupted, and MISSERR is set.

The program mustn't exceed 6ms for a full row between 2 erases. This is guaranteed by the sequence of 128 double words successively written with a clock system greater or equal to 8 MHz. An internal time-out counter counts 5.5ms when fast programming is set and stops the programming when the time-out is over. In this case, the FASTERR bit is set.

If an error occurs, programming is stopped and next double word to programmed is not programmed. Anyway, all previous double words have been properly programmed.

3.3.8 Programming Errors

Several kinds of errors are detected. In case of error, the Flash operation (programming or erasing) is aborted.

- **PROGERR: Programming Error**

  In standard programming: PROGERR is set if the word to write is not previously erased (except if the value to program is full zero).

- **SIZERR: Size Programming Error**

  In standard programming or in fast programming: only double word can be programmed and only 32-bit data can be written. SIZERR is set if a byte or a half-word is written.

- **PGAERR: Alignment Programming Error**

  PGAERR is set if one of the following conditions occurs:

  – In standard programming: the first word to be programmed is not aligned with a double word address, or the second word doesn't belong to the same double word address.
  – In fast programming: the data to program doesn't belong to the same row than the previous programmed double words, or the address to program is not greater than the previous one.

- **PGSERR: Programming Sequence Error**

  PGSERR is set if one of the following conditions occurs:

  – In the standard programming sequence or the fast programming sequence: Data is written when PG and FSTPG are cleared.
  – In the standard programming sequence or the fast programming sequence: MER0, MER1, and SER are not cleared when PG or FSTPG is set.
  – In the fast programming sequence: the Mass erase is not performed before setting the FSTPG bit.
  – In the mass erase sequence: PG, FSTPG, and SER are not cleared when MER0 or MER1 is set.
  – In the sector erase sequence: PG, FSTPG, MER0, and MER1 are not cleared when SER is set.

  PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR or PGSERR is set due to a previous programming error.

– When DBANK = 0, in the case that only either MER0 or MER1 is set, PGSERR is set (bank mass erase is not allowed).

- **WRPERR: Write Protection Error**

  WRPERR is set if one of the following conditions occurs:

  – Attempt to program or erase in a write protected area (WRP) or in a PCROP area or in a Securable memory area.
  – Attempt to perform a bank erase when one sector or more is protected by WRP or PCROP.
  – The debug features are connected or the boot is executed from SRAM or from System Flash when the Read Protection (RDP) is set to Level 1.
  – Attempt to modify the option bytes when the Read Protection (RDP) is set to Level 2.

- **MISSERR: Fast Programming Data Miss Error**

  In fast programming: all the data must be written successively. MISSERR is set if the previous data programming is finished and the next data to program is not written yet.

- **FASTERR: Fast Programming Error**

  In fast programming: FASTERR is set if one of the following conditions occurs:

  – When FSTPG bit is set for more than 7ms which generates a time-out detection.
  – When the fast programming has been interrupted by a MISSERR, PGAERR, WRPERR, or SIZERR.

- **RDERR: Reading Error**

  If attempt to read PCROP area, RDERR is set.

- **OPTVERR: Option Byte Valid Error**

  If option bytes configuration is invalid, OPTVERR is set.

If an error occurs during a program or erase operation, one of the following error flags is set in the FMC_SR register:

PROGERR, SIZERR, PGAERR, PGSERR, MISSERR (Program error flags), WRPERR (Protection error flag)

In this case, if the error interrupt enable bit ERRIE is set in the Flash status register (FMC_SR), an interrupt is generated and the operation error flag OPERR is set in the FMC_SR register.

NOTE: If several successive errors are detected (for example, in case of DMA transfer to the Flash memory), the error flags cannot be cleared until the end of the successive write requests.

3.3.9 Read-While-Write (RWW) (Only Available in Dual-bank Mode)

The dual bank mode is available only when the DBANK option bit is set, allowing Read-While-Write (RWW) operations. This feature allows performing a read operation from one bank while an erase or program operation is performed to another bank.

NOTE: Write-While-Write operations are not allowed. As an example, It is not possible to perform an erase operation on one bank while programming the other one.

Read from bank while sector/mass erasing in another bank (and vice versa)

While executing a program code from bank 0, it is possible to perform a sector erase operation on bank 1 (and vice versa). Follow the procedure below:

1. First check MER0 = 1 or MER1 = 1 to know current bank in executing.
2. Check that no Flash memory operation is ongoing by checking the BUSY bit in the FMC_SR (BUSY is active when erase/program operation is ongoing in bank 0 or bank 1).
3. For sector erase, set SER bit, SNB bits to select the sector, and BKER to select the bank in the FMC_CR.
4. For mass erase, set MER0 or MER1 (rely on step 1 result) in the FMC controller register (FMC_CR).
5. Set the STRT bit in the FMC_CR register.
6. Wait for the BUSY bit to be cleared (or use the EOP interrupt).

Read from bank while programming in another bank (and vice versa)

While executing a program code from bank 0, it is possible to perform a program operation on the bank 1 (and vice versa). Follow the procedure below:

1. Initially, verify whether MER0 = 1 or MER1 = 1 to identify the bank currently in execution.
2. Check that no Flash memory operation is ongoing by checking the BUSY bit in the FMC_SR (BUSY is active when erase/program operation is on-going on bank 0 or bank 1).
3. Set the PG bit in the FMC_CR.
4. Perform the data write operations at the desired address memory inside the main memory block.
5. Wait for the BUSY bit to be cleared (or use the EOP interrupt).

3.3.10 Option Bytes

The option bytes 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 (refer to Option Bytes Programming).

A double word is split up as follows in the option bytes:

Table 3-3 Option Byte Format

63-56	55-48	47-40	39-32	31-24	23-16	15-8	7-0
Complemented option byte 3	Complemented option byte 2	Complemented option byte 1	Complemented option byte 0	Option byte3	Option byte2	Option byte1	Option byte0

The option bytes can be read from the memory locations listed in Table 3-4 or from the option byte registers.

Table 3-4 TPS32M51xx Option Byte Organization

Address	[63:32]	[31:24]	[23:16]	[15:8]	[7:0]
0x0FFE8000	~[31:0]	USER OPT			RDP
0x0FFE8008	~[31:0]	8'h0	{7'h0, PCROP1STRT[16:0]}		
0x0FFE8010	~[31:0]	{PCROPRDP, 7'h0}	{7'h0, PCROP1END[16:0]}		
0x0FFE8018	~[31:0]	8'h0	WRP1AEND[7 :0]	8'h0	WRP1ASTRT[7:0]
0x0FFE8020	~[31:0]	8'h0	WRP2AEND[7 :0]	8'h0	WRP2ASTRT[7:0]
0x0FFE8028	~[31:0]	{15'h0, BOOTLOCK, 16'h0}			
0x0FFE8030	~[31:0]	unused			

0x0FFE8038	~[31:0]	8'h0	{7'h0, PCROP2STRT[16:0]}		
0x0FFE8040	~[31:0]	8'h0	{7'h0, PCROP2END[16:0]}		
0x0FFE8048	~[31:0]	8'h0	WRP1BEND[7 :0]	8'h0	WRP1BSTRT[7:0]
0x0FFE8050	~[31:0]	8'h0	WRP2BEND[7 :0]	8'h0	WRP2BSTRT[7:0]
0x0FFE8058	~[31:0]	Unused			

3.3.10.1 Option Bytes Programming

Option Byte Unlock

After reset, the options-related bits in the FMC_CR are write-protected. To run any operation on the option bytes sector, the option lock bit OPTLOCK in the FMC_CR must be cleared. The following sequence is used to unlock this register:

1. Unlock the FMC_CR with the LOCK clearing sequence (refer to Unlock FMC_CR).
2. Write OPTKEY1 = 0x0819 2A3B in the FMC_OPTKEYR register.
3. Write OPTKEY2 = 0x4C5D 6E7F in the FMC_OPTKEYR register.

The user options can be protected against unwanted erase/program operations by setting the OPTLOCK bit by software.

NOTE: If the LOCK is set by software, the OPTLOCK is automatically set as well.

Option Byte Modify

The option bytes are programmed differently from a main memory user address. To modify the user options value, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BUSY bit in the FMC_SR.
2. Unlock option bytes as the sequence described above.
3. Write the desired options value in the options register.
4. Set the options start bit (OPTSTRT) in the FMC_CR.
5. Wait for the BUSY bit to be cleared.

NOTE: Any modification of the value in one option is automatically performed by erasing whole user option bytes sector first and then programming all the option bytes with the values contained in the Flash option registers.

3.3.11 Option Bytes Loading

All new options are updated into the Flash, but they are not immediately applied to the system. To apply these options and have them take effect on the system, follow the procedure outlined below:

1. Set OBLLAUNCH bit in the FMC_CR.
2. A power reset (BOR reset or exit from standby/shutdown modes).

During option bytes loading, the options are ready by double word. If the word and its complement are matching. The option word/byte is executed. If comparison is failed between the option and its complement, a status it OPTVERR is set. Mismatch values are forced into the option registers.

- For USR OPT option, the value of mismatch is all options at ‘1’
- For WRP option, the value of mismatch is the default value “No protection”
- For RDP option, the value of mismatch is the default value “Level 1”

- For PCROP, the value of mismatch is “all memory protected”

3.3.12 Dual-bank Mode Activating/De-activating (If Dual-bank Mode Support)

Activating Dual-bank Mode

When switching from one Flash mode to another (for example, from single bank to dual bank), it is recommended to execute the code from the SRAM or use the bootloader. To avoid reading corrupted data from the Flash when the memory organization is changed, any access (either CPU or DMAs) to Flash memory should be avoided before reprogramming.

- Disable instructions or prefetch if they are enabled.
- Set the DBANK option bit and clear all the WRP write protection (follow user option modification and option bytes loader procedure).
  - Once OBLLAUNCH is done with DBANK = 1, perform a mass erase.
  - Start a new programming of code in 64-bit mode with DBANK = 1 memory mapping.
  - Set the new WRP/PCROP with DBANK = 1 scheme if needed.
  - Set PRFTEN.

De-activating Dual-bank Mode

When switching from one Flash mode to another (for example, from dual bank to single bank), it is recommended to execute the code from the SRAM or use the bootloader. To avoid reading corrupted data from the Flash when the memory organization is changed, any access (either CPU or DMAs) to Flash memory should be avoided before reprogramming.

- Disable instructions or prefetch if they are enabled.
- Clear the DBANK option bit, and all WRP write protection (follow user option modification and option bytes loader procedure).
  - Once OBLLAUNCH is done with DBANK = 0, perform a mass erase.
  - Start a new programming of code in 128-bit mode with DBANK = 0 memory mapping.
  - Set the new WRP/PCROP with DBANK = 0 scheme if needed.
  - Set PRFTEN if needed.

3.4 Flash Memory Protection

The Flash main memory can be protected against external accesses with the below methods:

- Read Protection (RDP)
- Proprietary Code Readout Protection (PCROP), the protection granularity is double word (64 bits) in dual bank mode or four words (128 bits) in single bank mode.
- Write Protection (WRP), the protection granularity is one sector (4 KB in dual bank mode or 8 KB in single bank mode).

3.4.1 Read Protection (RDP)

The Read Protection (RDP) is activated by setting the RDP field in user option byte register and then applying an OBL reset to reload the new RDP option byte. The read protection protects the Flash main memory and the option bytes.

NOTE: If the read protection is set while the debugger is still connected through JTAG/SWD, apply a Power-on Reset (POR) instead of a system reset.

There are three levels of read protection from no protection (level 0) to maximum protection or no debug (level 2).

The Flash memory is protected when the RDP option byte and its complement contain the pair of values shown in Table 3-5.

Table 3-5 Flash Memory Read Protection (RDP) Status

RDP Byte Value	RDP Complement Value	Read Protection Level
0xAA	0x55	Level 0 (production value)
Any value except 0xAA or 0xCC	Any value (not necessarily complementary) except 0x55 and 0x33	Level 1
0xCC	0x33	Level 2

The system memory area is always accessible for reading, regardless of the protection level. However, it is never accessible for program or erase operations.

Level 0: No Protection

Operations such as reading, programming, and erasing can be performed on the Flash main memory area and option bytes.

Level 1: Read Protection

This is the default protection level when RDP option byte is erased. It is defined as well when RDP value is at any value different from 0xAA and 0xCC, or even if the complement is not correct.

- **User mode**: Code executing in user mode (boot Flash) can access Flash main memory and option bytes with all operations.
- **Debug, boot RAM and bootloader modes**: In debug mode or when code is running from boot RAM or bootloader, the Flash main memory is totally inaccessible. In these modes, a read or write access to the Flash generates a bus error and a Hard Fault interrupt.

CAUTION

- If Level 1 is configured and no PCROP area is defined, the PCROPRDP bit must be set to 1. This will result in a full mass erase when the RDP level is decreased from Level 1 to Level 0.
- If Level 1 is configured and a PCROP area is defined, and the user code needs to be protected by RDP but not by PCROP, it should not be placed in a sector containing a PCROP area.

Level 2: No debug

In this level, the protection level 1 is guaranteed. In addition, the debug port, the boot from RAM (boot RAM mode) and the boot from System memory (boot loader mode) are no more available. In user execution mode (boot FLASH mode), all operations are allowed on the Flash Main memory. On the contrary, only read operations can be performed on the option bytes.

Option bytes cannot be programmed nor erased. Thus, the level 2 cannot be removed at all: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag WRPERR is set in the FMC_SR register and an interrupt can be generated.

NOTE: The debug feature is also disabled under reset. 3PEAK is not able to perform analysis on defective parts on which the level 2 protection has been set.

Changing the Read protection level

It is easy to move from level 0 to level 1 by changing the value of the RDP byte to any value (except 0xCC). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 either directly from level 0 or from level 1. Once in level 2, it is no more possible to modify the Read protection level.

When the RDP is reprogrammed to the value 0xAA to move from Level 1 to Level 0, a mass erase of the Flash main memory is performed by hardware if PCROPRDP is set in the Flash PCROP1 end address register (FMC_PCROP1ER). The user options are set to their previous values copied from FMC_OPTR. WRP and PCROP are disable. The OTP area is not affected by mass erase and remains unchanged.

If the bit PCROPRDP is cleared in the FMC_PCROP1ER, the full mass erase is replaced by a partial mass erase that is successive sector erases in the bank where PCROP is active, except for the sectors protected by PCROP. This is done to keep the PCROP code. If PCROP is active for both banks, both banks are erased by sector erases.

Only when both banks are erased, options are re-programmed with their previous values. This is also true for FMC_PCROPxSR and FMC_PCROPxER registers (x=1,2).

NOTE: Full mass erase or partial mass erase is performed only when Level 1 is active and Level 0 requested. When the protection level is increased (0->1, 1->2, 0->2) there is no mass erase.

To validate the protection level change, the option bytes must be reloaded through the OBLLAUNCH bit in Flash control register.

Table 3-6 Access Status vs. Protection Level and Execution Modes

Area	Protection Level	User Execution (Boot from Flash)			Debug/Boot from RAM/Boot from Loader (1)		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No (3)
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory (2)	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes	1	Yes	Yes (3)	Yes	Yes	Yes (3)	Yes
	2	Yes	No	No	N/A	N/A	N/A
OTP	1	Yes	Yes (4)	N/A	No	No	N/A
	2	Yes	Yes (4)	N/A	N/A	N/A	N/A

(1) When the protection level 2 is active, the Debug port, the boot from RAM and the boot from system memory are disabled.

(2) The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.

(3) The Flash main memory is erased when the RDP option byte is programmed with all level protections disabled (0xAA).

(4) OTP can only be written once.

3.4.2 Proprietary Code Readout Protection (PCROP)

A part of the Flash memory can be protected against read and write from third parties. The protected area is execute-only: it can only be reached by the ARM STAR core C-AHB bus, as an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. Depending on the DBANK mode, it allows either to specify one PCROP zone per bank in dual bank mode or to specify two PCROP zones for all memory. An additional option bit (PCROPRDP) allows to select if the PCROP area is erased or not when the RDP protection is changed from Level 1 to Level 0.

Each PCROP area is defined by double 2 words or 4 words.

In single bank mode (DBANK = 0):

The PCROPx (x = 1,2) area is defined from the address: base address + [PCROPxSTRT x 16bytes] (included) to the address: base address + [(PCROPxEND+1) x 16bytes] (excluded). The minimum PCROP area size is 2 x double-words (128 bits)

In dual bank mode (DBANK = 1)

The PCROPx (x = 1,2) area is defined from the address: bank “x” base address + [PCROPxSTRT x 8bytes] (included) to the address: bank “x” base address + [(PCROPxEND+1) x 8bytes] (exclud- ed). The minimum PCROP area size is 1 double- words (64 bits).

For example, to protect by PCROP from the address 0x0802 2880 (included) to the address 0x0803 0016 (included):

- if boot in Flash is done in Bank 0, FMC_PCROP1SR and FMC_PCROP1ER registers must be programmed with:
  - PCROP1STRT = 0x4510.
  - PCROP1END = 0x6002.
- If the two banks are swapped, the protection must apply to bank 1, and FMC_PCROP2SR and FMC_PCROP2ER register must be programmed with:
  - PCROP2STRT = 0x4510.
  - PCROP2END = 0x6002.

Any read access performed through the C-AHB bus (data access) to a PCROP protected area triggers the RDERR flag error.

Any PCROP protected address is also write protected and any write access to one of these address- es triggers WRPERR.

Any PCROP area is also erase protected. Consequently, any erase to a sector in this zone is impossible (including the sector containing the start address and the end address of this zone). Moreover, a software mass erase cannot be performed if one zone is PCROP protected.

For previous example, due to erase by sector, all sectors from sector 0x22 to 0x30 are protected in case of sector erase. (All addresses from 0x0802 2000 to 0x080 30FFF can’t be erased).

Deactivation of PCROP can only occurs when the RDP is changing from level 1 to level 0. If the user options modification tries to clear PCROP or to decrease the PCROP area, the options programming is launched but PCROP area stays unchanged. On the contrary, it is possible to increase the PCROP area.

When option bit PCROPRDP is cleared, when the RDP is changing from level 1 to level 0, Full Mass Erase is replaced by Partial Mass Erase to keep the PCROP area. In this case, PCROP1/2STRT and PCROP1/2END are also not erased.

NOTE: It is recommended to align PCROP area with sector granularity when using PCROPRDP, or to leave free the rest of the sector where PCROP zone starts or ends.

Table 3-7 PCROP Protection (1) (2)

PCROPx Registers Values (x=1,2)	PCROP Protection Area
PCROPxSTRT > PCROPxEND	No PCROP area
PCROPxSTRT ≤ PCROPxEND	The area between PCROPxSTRT and PCROPxEND is protected. It is possible to write: • PCROPxSTRT with a lower value • PCROPxEND with a higher value

(1) When DBANK = 1, the minimum PCROP area size is double words: PCROPxSTRT and PCROPxEND. When DBANK = 0, the minimum PCROP area size is 2x double words.

(2) When DBANK = 1, it is the user's responsibility to make sure no overlapping occurs on the PCROP zones.

3.4.3 Write Protection (WRP)

The user area in Flash memory can be protected against unwanted write operations. Depending on the DBANK option bit configuration, it allows either to specify:

- In single bank mode (DBANK = 0): Four write-protected (WRP) areas can be defined in each bank, with sector size (8 KB) granularity.
- In dual bank mode (DBANK = 1): Two write-protected (WRP) areas can be defined in each bank, with sector (4 KB) granularity.

Each area is defined by a start sector offset and an end sector offset related to the physical Flash bank base address. These offsets are defined in the WRP address registers:

- FMC_WRP1AR
- FMC_WRP1BR
- FMC_WRP2AR
- FMC_WRP2BR

Dual Bank Mode (DBANK = 1)

The bank "x" WRP "y" area (x = 1, 2 and y = A, B) is defined from the address: Bank "x" Base address + [WRPxySTRT x 0x1000bytes] (included) to the address: Bank "x" Base address + [(WRPxyEND+1) x 0x1000bytes] (excluded).

Single Bank Mode (DBANK = 0)

The WRPx "y" area (x = 1, 2 and y = A, B) is defined from the address: Base address + [WRPySTRT x 0x2000bytes] (included) to the address: Base address + [(WRPyEND+1) x 0x2000bytes] (excluded).

For example, to protect by WRP from 0x0806 2000 to 0x0807 0FFF (single bank):

- If boot in Flash is done in bank 0, FMC_WRP1AR register must be programmed with:
  - WRP1A_STRT = 0x62.
  - WRP2A_END = 0x70.

WRP1BSTRT and WRP1BEND in FMC_WRP1BR can be used instead (area B in Bank 0).

- If the two banks are swapped, the protection must apply to bank 1, and the FMC_WRP2AR register must be programmed with:
  - WRP2A_STRT = 0x62.
  - WRP2A_END = 0x70.

WRP2BSTRT and WRP2BEND in FMC_WRP2BR can be used instead (area B in Bank 1).

When WRP is active, it cannot be erased or programmed. Consequently, a software mass erase cannot be performed if one area is write-protected.

If an erase/program operation to a write-protected part of the Flash memory is attempted, the write protection error flag (WRPERR) is set in the FMC_SR register. This flag is also set for any write access to the PCROP area.

NOTE: To validate the WRP options, the option bytes must be reloaded by setting the OBLLAUNCH bit in the Flash control (FMC_CR) register. When DBANK = 0, the user should ensure no overlap in the WRP zone.

Table 3-8 WRP Protection

WRP Registers Values (x=1/2 y= A/B)	WRP Protection Area
WRPxySTRT = WRPxyEND	Sector WRPxy is protected.
WRPxySTRT > WRPxyEND	No WRP area.
WRPxySTRT < WRPxyEND	The sectors from WRPxySTRT to WRPxyEND are protected.

3.5 Forcing Boot from Flash Memory

To increase the security, the BOOTLOCK option bit of the FMC_BOOTR register allows forcing the system to boot from the Main Flash memory regardless the other boot options. It is always possible to set the BOOTLOCK bit. However, it is possible to reset it only when:

- RDP is set to Level 0.
- RDP is set to Level 1, while Level 0 is requested, and a full mass-erase is performed.

3.6 Interrupts

Table 3-9 Flash Interrupt Request

Interrupt Event	Event Flag	Event Flag/Interrupt Clearing Method	Interrupt Enable Control Bit
End of operation	EOP (1)	Write EOP = 1	EOPIE
Operation error	OPERR (2)	Write OPERR = 1	ERRIE
Read error	RDERR	Write RDERR = 1	RDERRIE
Program interrupt	PGINT (3)	Write PGINT = 1	PGIE

(1) EOP is set only if EOPIE is set.

(2) OPERR is set only if ERRIE is set.

(3) PGINT is set only if PGIE is set and 1 word (32 bits) will be written each time.

3.7 Registers

3.7.1 Register Address Map

Offset	Register Name	Register Description
0x0000	FMC_ACR	Flash access control register
0x0004	FMC_PDKEYR	Flash power-down key register
0x0008	FMC_KEYR	Flash key register
0x000c	FMC_OPTKEYR	Flash option key register
0x0010	FMC_SR	Flash status register
0x0014	FMC_CR	Flash control register
0x0018	FMC_PESR	Program and erase status register
0x0020	FMC_OPTR	Flash option register
0x0024	FMC_PCROP1SR	Flash PCROP1 start address register
0x0028	FMC_PCROP1ER	Flash PCROP1 end address register
0x002c	FMC_WRP1AR	Flash bank 0 WRP area A address register
0x0030	FMC_WRP1BR	Flash bank 0 WRP area B address register
0x0044	FMC_PCROP2SR	Flash PCROP2 start address register
0x0048	FMC_PCROP2ER	Flash PCROP2 end address register
0x004c	FMC_WRP2AR	Flash bank 1 WRP area A address register
0x0050	FMC_WRP2BR	Flash bank 1 WRP area B address register
0x0070	FMC_BOOTR	Flash boot lock register

3.7.2 Register Field Details

3.7.2.1 FMC_ACR

0x0000			Flash access control register											FMC_ACR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved						LVCTL	WCLKMODE	Reserved					WMODE		
Type	RO						WR	RW	RO					RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RCLKMODE	SLPPD	RUNPD	Reserved				PRFTEN	Reserved			RMODE				
Type	RW	RW	RW	RO				RW	RO			RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-10 Flash Access Control Register Description

Field	Name	Description
31:26	Reserved	Reserved
25	LVCTL	0: FMC low voltage mode (low-power run) disable (1.1 V) 1: FMC low voltage mode (low-power run) enable (1.0 V) **NOTE:** Typically, program and erase operation are not allowed when in low voltage mode. Additionally, the RMODEA requires attention when in low voltage mode (low-power run). The formula is Freq(WCLK) = Freq(IHS)/2^(WMODE) The software should configure the WMODE to ensure that Freq(system clk) >= 2*Freq(WCLK)
24	WCLKMODE	0: WCLK (IHS) autogating disable 1: WCLK (IHS) autogating enable

Field	Name	Description
23:18	Reserved	Reserved
17:16	WMODE	WCLK (WR clock) clock frequency divider 00: 8 MHz 01: 4 MHz 10: 2 MHz 11: 1 MHz **NOTE:** The WCLK should be lower than the system clock frequency. The formula is Freq(WCLK) = Freq(IHS)/2^(WMODE) The software should configure WMODE to make sure Freq(system clk) >= 2*Freq(WCLK)
15	RCLKMODE	0: Flash clock autogating enable 1: Flash clock autogating disable
14	SLPPD	Flash power-down mode during sleep or low-power sleep mode. This bit determines whether the Flash memory is in power-down or Idle mode when the device is in sleep or low-power sleep mode. 0: Flash in idle mode during sleep and low-power sleep modes 1: Flash in power-down mode during Sleep and Low-power sleep modes **NOTE:** The Flash must not be put in power-down while a program or an erase operation is ongoing.
13	RUNPD	Flash power-down mode during run or low-power run mode This bit is write-protected with FMC_PDKEYR. This bit determines whether the Flash memory is in power-down or idle mode when the device is in a run or low-power run mode. The Flash memory can be put in power-down mode only when the code is executed from RAM. The Flash must not be accessed when RUNPD is set. 0: Flash in idle mode 1: Flash in power-down mode

Field	Name	Description
		NOTE: The Flash must not be put in power-down while a program or an erase operation is ongoing.
12:9	Reserved	Reserved
8	PRFTEN	0: Prefetch disable 1: Prefetch enable
7:5	Reserved	Reserved
4:0	RMODE	System clock frequency These bits represent the frequency of the SYSCLK (system clock) period to the Flash access time. 00000: Equal or less than 8 MHz 00001: Equal or less than 16 MHz ... 10011: Equal or less than 160 MHz

3.7.2.2 FMC_PDKEYR

0x0004			Flash Power-down key register											FMC_PDKEYR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PDKEY															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PDKEY															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-11 Flash Power-down Key Register Description

Field	Name	Description
31:0	PDKEY	Flash key register The following values should be set consecutively to unlock the FMC_ACR-RUNPD bit: KEY1: 0x0415_1637 KEY2: 0xFAFB_FCFD

3.7.2.3 FMC_KEYR

0x0008			Flash key register											FMC_KEYR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-12 Flash Key Register Description

Field	Name	Description
31:0	KEY	Flash key register The following values should be set consecutively to unlock the FMC_CR register allowing program/erase operation: KEY1: 0x4567_0123 KEY2: 0XCDEF_89AB

3.7.2.4 FMC_OPTKEYR

0x000c			Flash option key register											FMC_OPTKEYR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	OPTKEY															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OPTKEY															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-13 Flash Option Key Register Description

Field	Name	Description
31:0	OPTKEY	Flash option byte key register The following values should be set consecutively to unlock the FMC_OPTR register allowing program/erase operation: KEY1: 0x0819_2a3b KEY2: 0x4C5D_6E7F

3.7.2.5 FMC_SR

0x0010			Flash status register											FMC_SR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved														PGINT	BUSY
Type	RO														WC	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OPTVERR	RDERR	Reserved				FASTERR	MISSERR	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Reserved	OPERR	EOP
Type	RC_W1	RC_W1	RO				RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RO	RC_W1	RC_W1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-14 Flash Status Register Description

Field	Name	Description
31:18	Reserved	Reserved
17	PGINT	0: Program interrupt raw status invalid 1: Program interrupt raw status valid
16	BUSY	This indicates that a Flash operation is in progress. This is set at the beginning of a Flash operation and reset when the operation finishes or when an error occurs.
15	OPTVERR	Set by hardware when the options read may not be the one configured by the user. If option hasn't been properly loaded, OPTVERR is set again after each system reset. Cleared by writing 1.
14	RDERR	Set by hardware when an address to be read through the C-bus data access belongs to a read-protected area of the Flash (PCROP protection). An interrupt is generated if RDERRIE is set in FMC_CR.

Field	Name	Description
		Cleared by writing 1.
13:10	Reserved	Reserved
9	FASTERR	Set by hardware when a fast programming sequence (activated by FSTPG) is interrupted due to an error (alignment, size, write protection or data miss). The corresponding status bit (PGAERR, SIZERR, WRPERR or MISSERR) is set at the same time. Cleared by writing 1.
8	MISSERR	In Fast programming mode, 32 double words must be sent to Flash successively, and the new data must be sent to the Flash logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time. Cleared by writing 1.
7	PGSERR	Set by hardware when write access to the Flash memory is performed by the code, while PG or FSTPG have not been set previously. Set also by hardware when PROGERR, SIZERR, PGAERR, WRPERR, MISSERR or FASTERR is set due to a previous programming error. Set also when trying to perform bank erase when DBANK = 0. Cleared by writing 1.
6	SIZERR	Set by hardware when the size of the access is a byte or half-word during a program or a fast program sequence. Only double word programming is allowed (consequently, word access). Cleared by writing 1.
5	PGAERR	Set by hardware when the data to program cannot be contained in the same 64-bit Flash memory row in case of standard programming, or if there is a change of sector during fast programming. Cleared by writing 1.
4	WRPERR	Set by hardware when an address to be erased/programmed belongs to a write-protected part (by WRP, PCROP or RDP level 1) of the Flash memory. Cleared by writing 1.

Field	Name	Description
3	PROGERR	Set by hardware when a double-word address to be programmed contains a value different from '0xFFFF FFFF' before programming, except if the data to write is '0x0000 0000'. Cleared by writing 1.
2	Reserved	Reserved
1	OPERR	Set by hardware when a Flash memory operation (program/erase) completes unsuccessfully. This bit is set only if error interrupts are enabled (ERRIE = 1). Cleared by writing 1.
0	EOP	Set by hardware when one or more Flash memory operations (programming/erase) have been completed successfully. This bit is set only if the end-of-operation interrupts are enabled (EOPIE = 1). Cleared by writing 1.

3.7.2.6 FMC_CR

0x0014			Flash control register											FMC_CR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	LOCK	OPTLOCK	PGIE	DMAEN	OBLLAUNCH	RDERRIE	ERRIE	EOPIE	Reserved					FSTPG	OPTSTRT	STRT
Type	RS	RS	RW	RW	RW	RW	RW	RW	RO					RW	RS	RS
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MER1	NVR	Reserved		BKER	SNB								MER0	SER	PG
Type	RW	RW	RO		RW	RW								RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-15 Flash Control Register Description

Field	Name	Description
31	LOCK	This bit is set only. When set, the FMC_CR register is locked. It is cleared by hardware after detecting the unlock sequence. In case of an unsuccessful unlock operation, this bit remains set until the next system reset.
30	OPTLOCK	This bit is set only. When set, all bits concerning user option in the FMC_CR register and so option sector are locked. This bit is cleared by hardware after detecting the unlock sequence. The LOCK bit must be cleared before doing the unlock sequence for the OPTLOCK bit. In case of an unsuccessful unlock operation, this bit remains set until the next reset.
29	PGIE	0: Program interrupt disable 1: Program interrupt enable
28	DMAEN	0: Program by DMA disable

Field	Name	Description
		1: Program by DMA enable
27	OBLLAUNCH	When set to 1, this bit forces the option byte reloading. This bit is cleared only when the option byte loading is complete. It cannot be written if OPTLOCK is set. 0: Option byte loading complete 1: Option byte loading requested
26	RDERRIE	This bit enables the interrupt generation when the RDERR bit in the FMC_SR is set to 1. 0: PCROP read error interrupt disabled 1: PCROP read error interrupt enabled
25	ERRIE	This bit enables the interrupt generation when the OPERR bit in the FMC_SR is set to 1. 0: OPERR error interrupt disabled 1: OPERR error interrupt enabled
24	EOPIE	This bit enables the interrupt generation when the EOP bit in the FMC_SR is set to 1. 0: EOP Interrupt disabled 1: EOP Interrupt enabled
23:19	Reserved	Reserved
18	FSTPG	0: Fast programming disabled 1: Fast programming enabled
17	OPTSTRT	This bit triggers an options operation when set. This bit is set only by software and is cleared when the BSY bit is cleared in FMC_SR.
16	STRT	This bit triggers an erase operation when set. If MER1, MER2 and SER bits are reset, and the STRT bit is set, unpredictable behavior may occur without generating any error flag. This condition should be forbidden. This bit is set only by software and is cleared when the BSY bit is cleared in FMC_SR.
15	MER1	This bit triggers the bank 1 mass erase (all bank 1 user sectors) when set.
14	NVR	Bank1 NVR sector erase enable. Only used for redundancy main sectors (64 KB) in bank1.

Field	Name	Description
13:12	Reserved	Reserved
11	BKER	Bank erase DBANK = 1 0: Bank 0 is selected for sector erase 1: Bank 1 is selected for sector erase DBANK = 0 Reserved, must be kept clear.
10:3	SNB	Sector number These bits select the sector to erase: 00000000: Sector 0 00000001: Sector 1 ... 11111111: Sector 255
2	MER0	Bank 0 Mass erase This bit triggers the bank 0 mass erase (all bank 0 user pages) when set.
1	SER	Sector erase 0: Sector erase disabled 1: Sector erase enabled
0	PG	Programming 0: Flash programming disabled 1: Flash programming enabled

3.7.2.7 FMC_PESR

0x0018			Program and erase status register											FMC_PESR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															FERFLAG
Type	RO															WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-16 Program and Erase Status Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	FERFLAG	1: When mass/sector erase is performed once, this bit will be automatically set to 1 by the hardware. Once it is set to 1, the software can only clear this bit by writing 1 if all the fast program work is completed (as decided by the software). If FERFLAG is 0, and a fast program is being executed, then PGSERR assert. 0: Default value and no mass/sector erase have been done ever.

3.7.2.8 FMC_OPTR

0x0020			Flash option register											FMC_OPTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		NRSTMODE		NBOOT0	NSWBOOT0	Reserved		NBOOT1	DBANK	Reserved	SWAPBANK	WWDGSW	IWDGSTDBY	IWDGSTOP	IWDGSW
Type	RW	RO	RW		RW	RW	RO		RW	RW	RO	RW	RW	RW	RW	RW
Reset	0	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	NRSTLVL	RSTSHDW	RSTSTDBY	RSTSTOP	Reserved	BORLEV			RDP							
Type	RW	RW	RW	RW	RO	RW			RW							
Reset	0	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1

Table 3-17 Flash Option Register Description

Field	Name	Description
31:30	Reserved	Reserved
29:28	NRSTMODE	00: Reserved 01: Reset input only; a low level on the NRST pin generates a system reset, internal RESET is not propagated to the NSRT pin 11: Bidirectional reset; NRST pin configured in reset input/output mode (legacy mode)
27	NBOOT0	NBOOT0 option bit 0: NBOOT0 = 0 1: NBOOT0 = 1
26	NSWBOOT0	Software BOOT0 0: BOOT0 taken from the option bit NBOOT0 1: BOOT0 taken from PB8/BOOT0 pin

Field	Name	Description
25:24	Reserved	Reserved
23	NBOOT1	Boot configuration Together with the BOOT0 pin, this bit selects boot mode from the Flash main memory, SRAM1 or the system memory.
22	DBANK	0: Single bank mode with 128 bits of data read width 1: Dual bank mode with 64 bits of data This bit can only be written when PCROPA/B is disabled.
21	Reserved	Reserved
20	SWAPBANK	DBANK is 1: 1: Boot from bank1 0: Boot from bank0 DBANK is 0: This bit is not used and always boots from bank0.
19	WWDGSW	Window Watchdog (WWDG) selection 0: Hardware WWDG 1: Software WWDG
18	IWDGSTDBY	Independent Watchdog (IWDG) counter freeze in standby mode 0: IWDG counter is running in standby mode 1: IWDG counter is frozen in standby mode
17	IWDGSTOP	Independent Watchdog (IWDG) counter freeze in stop mode 0: IWDG counter is running in stop mode 1: IWDG counter is frozen in stop mode
16	IWDGSW	Independent Watchdog (IWDG) selection 0: Hardware IWDG 1: Software IWDG
15	NRSTLVL	0: NRST as system reset (default)

Field	Name	Description
		1: NRST as POR reset (reset all digital register except RTC, disable all analog except RTC)
14	RSTSHDW	0: No reset generated when entering the shutdown mode 1: Reset generated when entering the shutdown mode
13	RSTSTDBY	0: No reset generated when entering the standby mode 1: Reset generated when entering the standby mode
12	RSTSTOP	0: No reset generated when entering the stop mode 1: Reset generated when entering the stop mode
11	Reserved	Reserved
10:8	BORLEV	These bits contain the VDD supply level threshold that activates/releases the reset.
7:0	RDP	0xAA: Level 0, read protection not active 0xCC: Level 2, chip read protection active Others: Level 1, memories read protection active **NOTE:** It is important to be cautious when configuring PCROPRDP in level 1.

3.7.2.9 FMC_PCROP1SR

0x0024			Flash PCROP1 Start address register											FMC_PCROP1SR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															PCROP1STRT
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PCROP1STRT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-18 Flash PCROP1 Start Address Register Description

Field	Name	Description
31:17	Reserved	Reserved
16:0	PCROP1STRT	PCROP area start offset DBANK = 1 PCROP1STRT contains the first double word of the PCROP area for bank0. DBANK = 0 PCROP1STRT contains the first 2xdouble-word of the PCROP area for all memory.

3.7.2.10 FMC_PCROP1ER

0x0028			Flash PCROP1 End address register											FMC_PCROP1ER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PCROPRDP	Reserved														PCROP1END
Type	RS	RO														RW
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PCROP1END															
Type	RW															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 3-19 Flash PCROP1 End Address Register Description

Field	Name	Description
31	PCROPRDP	PCROP area preserved when the RDP level decreases This bit is set only. It is reset after a full mass erases due to a change of RDP from level 1 to level 0. 0: PCROP area is not erased when the RDP level is decreased from level 1 to level 0. 1: PCROP area is erased when the RDP level is decreased from level 1 to level 0 (full mass erase).
30:17	Reserved	Reserved
16:0	PCROP1END	Bank 0 PCROP area end offset DBANK = 1 PCROP1END contains the last double word of the bank 0 PCROP area. DBANK = 0 PCROP1END contains the last 2xdouble-word of the PCROP area for all the memory.

3.7.2.11 FMC_WRP1AR

0x002c			Flash bank 0 WRP area A address register											FMC_WRP1AR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								WRP1AEND							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								WRP1ASTRT							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-20 Flash Bank 0 WRP Area A Address Register Description

Field	Name	Description
31:24	Reserved	Reserved
23:16	WRP1AEND	WRP first area A end offset DBANK = 1 WRP1AEND contains the last page of WRP first area in bank0. DBANK = 0 WRP1AEND contains the last page of WRP first area for all memory.
15:8	Reserved	Reserved
7:0	WRP1ASTRT	WRP first area A start offset DBANK = 1 WRP1ASTRT contains the first page of WRP first area for bank0. DBANK = 0 WRP1ASTRT contains the first page of WRP first area for all memory.

3.7.2.12 FMC_WRP1BR

0x0030			Flash bank 0 WRP area B address register											FMC_WRP1BR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								WRP1BEND							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								WRP1BSTRT							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-21 Flash Bank 0 WRP Area B Address Register Description

Field	Name	Description
31:24	Reserved	Reserved
23:16	WRP1BEND	WRP second area B end offset DBANK = 1 WRP1BEND contains the last page of the WRP second area for bank0. DBANK = 0 WRP1BEND contains the last page of the WPR second area for all memory.
15:8	Reserved	Reserved
7:0	WRP1BSTRT	WRP second area B start offset DBANK = 1 WRP1BSTRT contains the last page of the WRP second area for bank0. DBANK = 0 WRP1BSTRT contains the last page of the WPR second area for all memory.

3.7.2.13 FMC_PCROP2SR

0x0044			Flash PCROP2 Start address register											FMC_PCROP2SR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															PCROP2STRT
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PCROP2STRT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-22 Flash PCROP2 Start Address Register Description

Field	Name	Description
31:17	Reserved	Reserved
16:0	PCROP2STRT	PCROP area start offset DBANK = 1 PCROP2STRT contains the first double-word of the PCROP area for bank 1. DBANK = 0 PCROP2STRT contains the first double-word PCROP area for all memory.

3.7.2.14 FMC_PCROP2ER

0x0048			Flash PCROP2 End address register											FMC_PCROP2ER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															PCROP2END
Type	RS	RO														RW
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PCROP2END															
Type	RW															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 3-23 Flash PCROP2 End Address Register Description

Field	Name	Description
31:17	Reserved	Reserved
16:0	PCROP2END	PCROP area end offset DBANK = 1 PCROP2END contains the last double-word of the PCROP area for bank1. DBANK = 0 PCROP2END contains the last 2x double-word of the PCROP area for all the memory.

3.7.2.15 FMC_WRP2AR

0x004c			Flash bank 1 WRP area A address register											FMC_WRP2AR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								WRP2AEND							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								WRP2ASTRT							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-24 Flash Bank 1 WRP Area A Address Register Description

Field	Name	Description
31:24	Reserved	Reserved
23:16	WRP2AEND	WRP first area A end offset DBANK = 1 WRP2AEND contains the last page of the WRP first area for bank1. DBANK = 0 WRP2AEND contains the last page of the WRP third area for all memory.
15:8	Reserved	Reserved
7:0	WRP2ASTRT	WRP first area A start offset DBANK = 1 WRP2ASTRT contains the first page of the WRP first area for bank1. DBANK = 0 WRP2ASTRT contains the first page of the WRP third area for all memory.

3.7.2.16 FMC_WRP2BR

0x0050			Flash bank 1 WRP area B address register											FMC_WRP2BR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								WRP2BEND							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								WRP2BSTRT							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Field	Name	Description
31:24	Reserved	Reserved
23:16	WRP2BEND	WRP second area B end offset DBANK = 1 WRP2BEND contains the last page of the WRP second area for bank1. DBANK = 0 WRP2BEND contains the last page of the WRP fourth area for all memory.
15:8	Reserved	Reserved
7:0	WRP2BSTRT	WRP second area B start offset DBANK = 1 WRP2BSTRT contains the first page of the WRP second area for bank1. DBANK = 0 WRP2BSTRT contains the first page of the WRP second area for all memory.

3.7.2.17 FMC_BOOTR

0x0070			Flash boot lock register											FMC_BOOTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															BOOTLOCK
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3-25 Flash Boot Lock Register

Field	Name	Description
31:17	Reserved	Reserved
16	BOOTLOCK	It is used to force boot from the user Flash area. 0: Boot based on the pad/option bit configuration 1: Boot forced from Main Flash memory
15:0	Reserved	Reserved

Power Management Unit (PMU)

This chapter provides details of the Power Management Unit (PMU).

Topics: **Page**

4.1 Introduction

The Power Management Unit (PMU) is responsible for managing all functions related to the power supply and its supervision for the device. Its main features include:

- Generation of voltage for the device's internal core logic (V_{CORE}).
- Provision of several mechanisms and reset sources for the voltage applied to the device.
- Management of multiple power-saving modes.

4.2 Features

- Wide supply voltage (V_{DD}) range: 1.71 V to 3.6 V. Refer to the device specification for more details on the range.
- Generation of voltage for the device's internal core logic (V_{CORE}).
- Internal regulator control during device operation mode transition.
- Power-on Reset (POR) and Brownout Reset (BOR) generation.
- Supply voltage supervisor (SVS).

4.3 Functional Description

The PMU uses integrated low-dropout voltage regulators (LDOs) to generate a secondary core voltage (V_{CORE}) from the primary voltage (V_{DD}) applied to the device. In general, V_{CORE} supplies the internal digital circuit, including the CPU, memories, and digital modules. V_{DDA} supplies analog modules such as ADC, DAC, CMP, and so on. V_{REF+} provides the reference voltage. Some devices only provide V_{DD} Pin to maximize pin usage. For more detailed information, please refer to the device specification.

An integrated internal bandgap circuit provides an internal voltage reference for maintaining the output voltage of V_{CORE}/$V_{STANDBY}$/V_{REF}. POR and BOR circuits are included to generate reset signals and monitor the power supply. Figure 4-1 illustrates the various components and their connections within the PMU.

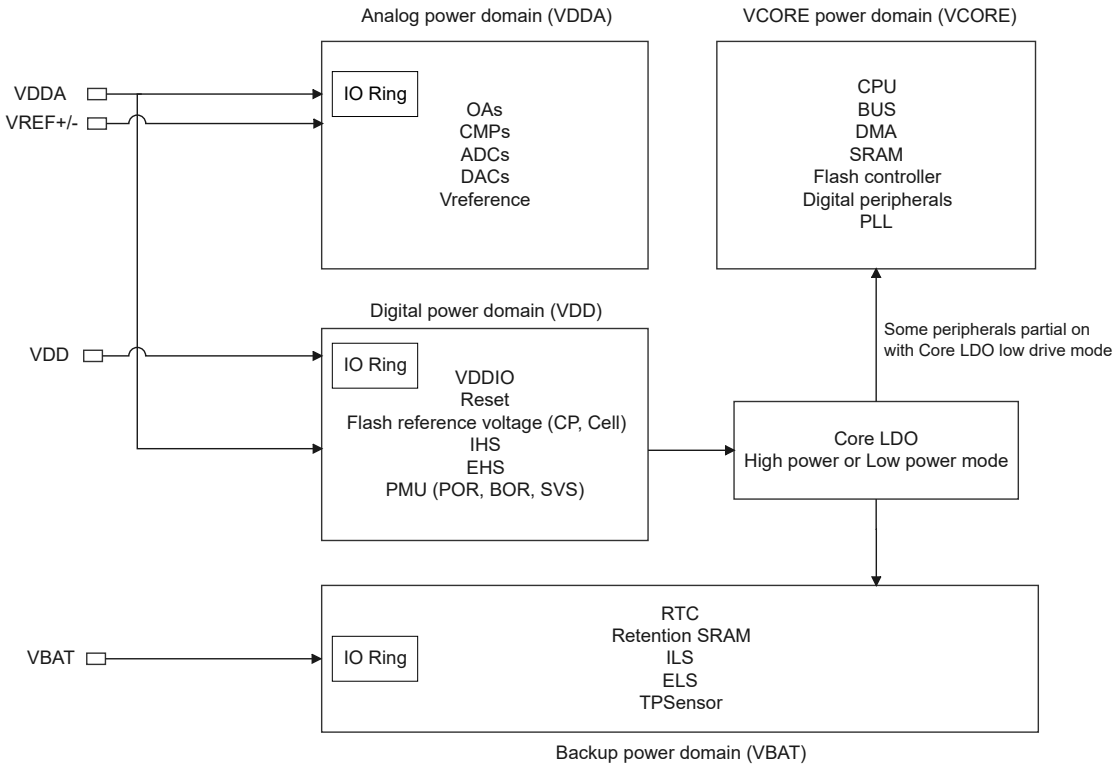


Figure 4-1 PMU Block Diagram

4.3.1 Internal Regulators

There are two internal LDO regulators known as LDO_CORE and LDO_RTC. LDO_CORE supports two different load settings: high power (1.1 V) and low power (1.0 V). These different LDO stages and load settings are provided to optimize power consumption.

The LDO_CORE is designed to support the run, sleep and stop modes, while LDO_RTC is designed for the standby1, standby2, and shutdown modes. The hardware automatically controls the LDO stages and load settings based on the device's different operation modes.

4.3.2 Power-On Reset (POR)/Power-Down Reset (PDR)/Brownout Reset (BOR)

The device features an integrated Power-on Reset (POR)/Power-down Reset (PDR) along with a Brown-out Reset (BOR) circuitry. The BOR is active in all power modes and cannot be disabled. During power-on, the BOR keeps the device under reset until the supply voltage V_{DD} reaches the specified V_{BORx} threshold.

When the V_{DD} drops below the BOR threshold, a device reset is triggered. Once the V_{DD} exceeds the V_{BOR} upper limit, the device reset is released, and the system can start. For more details on the BOR thresholds, refer to the electrical characteristics section in the datasheet.

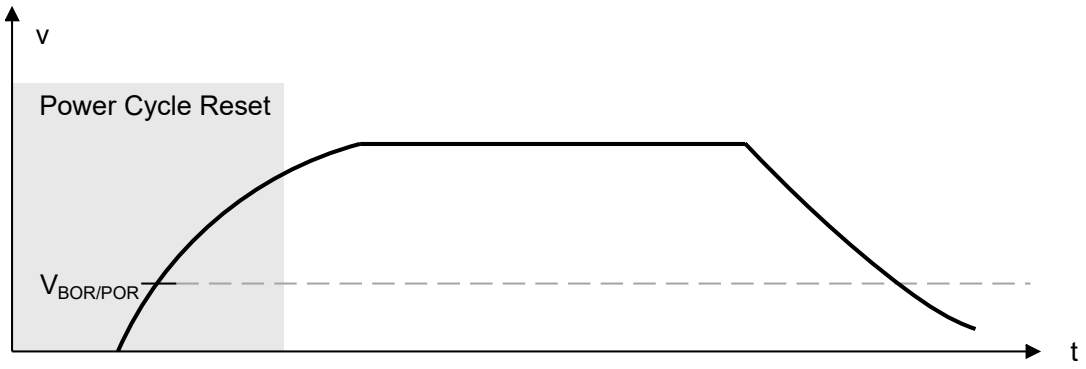


Figure 4-2 Power Supply Change and Resulting POR and BOR Actions

4.3.2.1 Supply Voltage Supervisor

An SVSO flag is available in the power status register 2 (PMU_SR2) to indicate whether V_{DD} is higher or lower than the SVS threshold. This event is internally connected to the EXTI line 16 and can generate an interrupt if enabled through the EXTI registers. The SVS output interrupt can be generated when V_{DD} drops below the SVS threshold.

4.3.3 Operation Modes

By default, the device is in run mode after a power reset or system reset. There are several low-power modes available to save power when the CPU does not need to be continuously running. Users can select the optimal balance mode, which provides low-power consumption, short startup/wake-up time and available peripherals.

4.3.3.1 Run Mode

The CPU operates in run mode, and all peripherals can also function in this mode to achieve optimal computational performance.

In run mode, the speed of the system clocks (SYSCLK, HCLK, PCLK) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down the peripherals for lower power consumption.

In run mode, the HCLK and PCLK for individual peripherals and memories can be stopped anytime to reduce power consumption. To further minimize the power consumption in sleep mode, the peripheral clocks can be disabled before executing the WFI or WFE instructions.

The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

Low-Power Run Mode (LP Run)

To further reduce consumption when the system is in run mode, the regulator can be configured in low-power mode. In this mode, the CPU frequency should not exceed 48 MHz. Refer to the datasheet for more details on voltage regulator and peripherals operating conditions.

Entering the Low-Power Run Mode

1. (Optional) If desired, Jump into SRAM code and enter Flash Deep Power Down (DPD) mode by setting the RUNPD bit in Flash access control register (FMC_ACR).
2. Reduce the CPU clock frequency to a value below 48 MHz.
3. Set the LVCTL bit in the FMC_ACR register for right Flash access timing before entering low-power run mode.
4. Switch the regulator to low-power mode by setting the LPR bit in the PMU_CR00 register.
5. Wait until the LPF bit is cleared in the PMU_SR2 register, indicating that the regulator adjustment is done.

For detailed instructions on how to enter the low-power run mode, please refer to Table 4-1.

Exiting the Low-Power Run Mode

1. Switch the regulator to the main mode by clearing the LPR bit in the PMU_CR00 register.
2. Wait until the LPF bit is cleared in the PMU_SR2 register, indicating that the regulator adjustment is done.
3. Clear the LVCTL bit in the FMC_ACR register for the right access timing when in normal run mode.

4. Increase the CPU clock frequency to the desired value.

For detailed instructions on how to exit the low-power run mode, please refer to Table 4-1.

Table 4-1 Low-Power Run

Low-Power Run Mode	Description
Mode Entry	Decrease the CPU clock frequency below 48 MHz FMC_ACR[LVCTL] = 1 LPR = 1 Wait until LPF = 0
Mode Exit	LPR = 0 Wait until LPF = 0 FMC_ACR[LVCTL] = 0 Increase the CPU clock frequency
Wakeup Latency	Regulator wakeup time from low-power mode

4.3.3.2 Low-Power Modes

Entering the Low-Power Mode

Low-power modes can be entered by the MCU through one of the following ways:

- Executing the Wait for Interrupt (WFI) or Wait for Event (WFE) instructions.

  It is important to note that entering the low-power mode through WFI or WFE is only executed if there are no pending interrupts or events.
- When the SLEEPONEXIT bit in the CMStar with the FPU system control register is set upon returning from ISR.

Exiting the Low-Power Mode

From sleep modes and stop modes, the MCU exits low-power mode depending on the way the low-power mode was entered:

- If the WFI instruction or return from ISR was used to enter the low-power mode, any peripheral interrupt acknowledged by the NVIC could wake up the device.

- If the WFE instruction is used to enter the low-power mode, the MCU exits the low-power mode as soon as an event occurs. The wakeup event can be generated either by:
  - NVIC IRQ interrupt
    - When SEVONPEND = 0 in the CMStar with FPU system control register: by enabling an interrupt in the peripheral control register and the NVIC.

      When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) need to be cleared.

      Only NVIC interrupts with sufficient priority wakeup and interrupts the MCU.
    - When SEVONPEND = 1 in the CMStar with FPU system control register: by enabling an interrupt in the peripheral control register and optionally in the NVIC.

      When the MCU resumes from WFE, the peripheral interrupt pending bit and, when enabled, the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) need to be cleared.

      All NVIC interrupts wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with sufficient priority wakeup and interrupt the MCU.
  - Event

    This is done by configuring an EXTI line in event mode. When the CPU resumes from WFE, there is no need to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit, since the pending bits associated with the event line are not set. However, it might still be necessary to clear the interrupt flag in the peripheral.

From standby1, standby2, and shutdown modes, the MCU exits low power mode by means of an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins, or when an RTC event occurs.

After waking up from standby1, standby2, or shutdown mode, program execution restarts in the same way as after a reset. This includes boot pin sampling, option bytes loading, and fetching the reset vector.

4.3.3.2.1 Sleep Mode

I/O States in Sleep Mode

In sleep mode, all I/O pins keep the same state as in run mode.

Entering the Sleep Mode

The sleep mode is entered according to Entering Low Power Mode, when the SLEEPDEEP bit in the CMStar with FPU system control register is clear.

Refer to Table 4-2 for details on how to enter the sleep mode.

Exiting the Sleep Mode

The sleep mode is exited according to Exiting Low Power Mode.

Refer to Table 4-2 for more details on how to exit the sleep mode.

Table 4-2 Sleep Mode

Sleep-Now Mode	Description
Mode Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: • SLEEPDEEP = 0 • No interrupt (for WFI) or event (for WFE) is pending Refer to the CMStar with FPU system control register.
	On return from ISR while: • SLEEPDEEP = 0 • SLEEPONEXIT = 1 • No interrupt is pending Refer to the CMStar with FPU system control register.
Mode Exit	If WFI or return from ISR was used for entry Interrupt: Refer to related device datasheet for series vector table. If WFE was used for entry and SEVONPEND = 0: Wakeup event: Refer to Wakeup Event Management. If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC (refer to datasheet for series vector table) or wakeup event (refer to Wakeup Event Management).
Wakeup Latency	None

4.3.3.2.2 Low-Power Sleep Mode (LP Sleep)

I/O States in Low-Power Sleep Mode

In low-power sleep mode, all I/O pins keep the same state as in run mode.

Entering the Low-Power Sleep Mode

The low-power sleep mode is entered from low-power run mode according to Entering the Low-Power Mode when the SLEEPDEEP bit in the CMStar with FPU system control register is clear.

Refer to Table 4-3 for details on how to enter the low-power sleep mode.

Exiting the Low-Power Sleep Mode

The low-power sleep mode is exited according to Exiting the Low-Power Mode. When exiting the Low-power sleep mode by issuing an interrupt or an event, the MCU is in low-power run mode. Refer to Table 4-3 for details on how to exit the low-power sleep mode.

Table 4-3 Low-Power Sleep

Low-Power Sleep-Now Mode	Description
Mode Entry	Low-power sleep mode is entered from the low-power run mode. WFI (Wait for Interrupt) or WFE (Wait for Event) while: • SLEEPDEEP = 0 • No interrupt (for WFI) or event (for WFE) is pending Refer to the CMStar with FPU system control register.
	Low-power sleep mode is entered from the low-power run mode. On return from ISR while: • SLEEPDEEP = 0 • SLEEPONEXIT = 1 • No interrupt is pending Refer to the CMStar with FPU system control register.

Low-Power Sleep-Now Mode	Description
Mode Exit	If WFI or return from ISR was used for entry Interrupt: refer to related device datasheet for series vector table. If WFE was used for entry and SEVONPEND = 0: Wakeup event: Refer to Wakeup Event Management. If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: Refer to related device datasheet for series vector table. Wakeup event: Refer to Wakeup Event Management. After exiting the low-power sleep mode, the MCU is in low-power run mode.
Wakeup Latency	None

4.3.3.2.3 Stop Mode

The stop mode is based on the CMStar deep sleep mode combined with the peripheral clock gating. The voltage regulator is configured in low-power regulator mode. In stop mode, all clocks in the V_{CORE} domain are stopped; the PLL, the IHS and the EHS oscillators are disabled. The BOR is always available in stop mode.

I/O States in Stop Mode

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

Entering the Stop Mode

The stop mode is entered according to entering low power mode when the SLEEPDEEP bit in the CMStar system control register is set.

Refer to Table 4-4 for details on how to enter the stop mode. If Flash memory programming is ongoing, the stop mode entry is delayed until the memory 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. Refer to IWDG Functional Description.
- Real-time clock (RTC): This is configured by the RTCEN bit in the RTC domain control register (RCC_BDCR)
- Internal RC oscillator (ILS): This is configured by the ILSON bit in the RCC_BDCR.
- 32.768 kHz External Low-Speed oscillator (ELS): This is configured by the ELSON bit in the RCC_BDCR.

- SRAM in CORE domain： Five 64 KB and one 16 KB SRAM are all in retention state.

The CMP and TPSensor will consume power in the stop mode unless they are disabled before entering this mode.

Exiting the Stop Mode

The stop mode is exit according to entering low power mode. Refer to Table 4-4 for details on how to exit stop mode. When exiting stop mode by issuing an interrupt or a wakeup event, the IHS oscillator is selected as the system clock. When the voltage regulator operates in low-power mode, an additional startup delay is required when waking up from stop mode with IHS. When exiting the stop mode, the MCU is either in run mode or in low-power run mode if the bit LPR is set in the PMU_CR01.

Table 4-4 Stop Mode

Stop Mode	Description
Mode Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: • SLEEPDEEP bit is set in CMStar system control register • No interrupt (for WFI) or event (for WFE) is pending • LPMS = 1000 in PMU_CR01
	On return from ISR while: • SLEEPDEEP bit is set in CMStar system control register • SLEEPONEXIT = 1 • No interrupt is pending • LPMS = 1000 in PMU_CR01
Mode Exit	If WFI or Return from ISR was used for entry Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to the datasheet for the series vector table. If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to Wakeup Event Management.

Stop Mode	Description
	If WFE was used for entry and SEVONPEND = 1: Any EXTI Line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wakeup capability. Refer to the datasheet for the series vector table. Wakeup event: Refer to Wakeup Event Management.
Wakeup Latency	

4.3.3.2.4 Standby Mode

Standby1 Mode

The standby1 mode allows to achieve the lowest power consumption with TPSensor and SRAM2 retention. It is based on the CMStar with FPU deep sleep mode, with the LDO_CORE disabled. The PLL, the IHS, and the EHS oscillators are also switched off.

The SVS is always available in standby mode. The consumption is increased when SVS is enabled.

I/O States in Standby1 Mode

In the standby1 mode, the I/Os can be kept in analog state, except for PE7 and PE8.

PE7 and PE8, used for ELS, are also functional. 5 wakeup pins (WKUPx, x = 1, 2 ... 5) are available.

Entering the Standby1 Mode

The standby1 mode is entered according to Entering the Low Power Mode, when the SLEEPDEEP bit in the CMStar with FPU system control register is set.

Refer to Table 4-5 for details on how to enter standby1 mode.

In standby1 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. Refer to Functional Description in Independent Watchdog (IWDG).
- Real-time Clock (RTC): Configured by the RTCEN bit in the RTC domain control register (RCC_BDCR).
- Internal RC Oscillator (ILS): Configured by the ILSON bit in the RCC control/status register (RCC_CSR).
- External 32.768 kHz Oscillator (ELS): Configured by the ELSON bit in the RCC_BDCR.

- SVS: Configured by the SVSEN bit in PMU_CR03.

Exiting the Standby1 Mode

The standby1 mode is exit according to Entering the Low-Power Sleep Mode. The SB1F status flag in the power status register 1 (PMU_SR1) indicates that the MCU was in standby1 mode. All registers are reset after wakeup from Standby except for PMU_SR1, which indicates that the MCU was in standby1, standby2 or shutdown mode.

When exiting standby1 mode by issuing an interrupt or a wakeup event, the IHS8 oscillator is selected as a system clock.

When exiting the standby1 mode, the MCU is in run mode.

Refer to Table 4-5 for more details on how to exit standby1 mode.

Table 4-5 Standby1 Mode

Standby1 Mode	Description
Mode Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: • SLEEPDEEP bit is set in CMStar with FPU system control register • No interrupt (for WFI) or event (for WFE) is pending • LPMS = “0001” in PMU_CR01 • WUFx bits are cleared in PMU_SR1
	On return from ISR while: • SLEEPDEEP bit is set in CMStar with FPU system control register • SLEEPONEXIT = 1 • No interrupt is pending • LPMS = “0001” in PMU_CR01 • WUFx bits are cleared in PMU_SR1 • The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup or timestamp flags) is cleared.

Standby1 Mode	Description
Mode Exit	WKUPx pin edge, RTC event, external reset in NRST pin, IWDG Reset, SVS event, TPSensor wakeup，ELS CSS failure
Wakeup Latency	Longest wakeup time between: IHS8 wakeup time + LDO_CORE ready from disable + Flash wakeup time from standby1 mode

Standby2 Mode

The standby2 mode is the same as the standby1 mode except the switch_tsr is off. The Retention_SRAM16K and TPSensor are powered off.

Entering the Standby2 Mode

The standby2 mode is entered according to Entering Low Power Mode, when the SLEEPDEEP bit in the CMStar with FPU system control register is set. Refer to Table 4-6 for details on how to enter standby2 mode.

In standby2 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. Refer to Functional Description in Independent Watchdog (IWDG).
- Real-time Clock (RTC): Configured by the RTCEN bit in the RCC_BDCR.
- Internal RC Oscillator (ILS): Configured by the ILSON bit in the RCC_BDCR.
- External 32.768 kHz Oscillator (ELS): Configured by the ELSON bit in the RCC_BDCR.
- SVS: Configured by the SVSEN bit in PMU_CR03.

Exiting the Standby2 Mode

The standby2 mode is exit according to Entering Low Power Mode. The SB2F status flag in the PMU_SR1 indicates that the MCU was in standby2 mode. All registers are reset after wakeup from standby except for PMU_SR1.

When exiting standby2 mode by issuing an interrupt or a wakeup event, the IHS oscillator is selected as a system clock.

When exiting the standby2 mode, the MCU is in run mode.

Refer to Table 4-6 for more details on how to exit standby2 mode.

Table 4-6 Standby2 Mode

Standby2 Mode	Description
Mode Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: • SLEEPDEEP bit is set in CMStar with FPU system control register • No interrupt (for WFI) or event (for WFE) is pending • LPMS = “0010” in PMU_CR1 • WUFx bits are cleared in PMU_SR1
	On return from ISR while: • SLEEPDEEP bit is set in CMStar with FPU system control register • SLEEPONEXIT = 1 • No interrupt is pending • LPMS = “0010” in PMU_CR1 • WUFx bits are cleared in PMU_SR1 • The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup or timestamp flags) is cleared.
Mode Exit	WKUPx pin edge, RTC event, external Reset in NRST pin, IWDG Reset, SVS event
Wakeup Latency	Reset phase

4.3.3.2.5 Shutdown Mode

The Shutdown mode allows to achieve the lowest power consumption. The Retention-SRAM16K, the TPSensor and IWDG, and the SVS function are off.

I/O States in Shutdown Mode

In the shutdown mode, I/Os can be kept in analog state. However, PE7 and PE8, used for ELS, are also functional. 5 wakeup pins (WKUPx, x=1,2...5) are available.

Entering the Shutdown Mode

The shutdown mode is entered according to Entering the Low-Power Mode, when the SLEEPDEEP bit in the CMStar with FPU system control register is set. Refer to Table 4-7 for details on how to enter shutdown mode. In shutdown mode, the following features can be selected by programming individual control bits:

- Real-time Clock (RTC): Configured by the RTCEN bit in the RTC domain control register (RCC_BDCR).
- Internal RC Oscillator (ILS): Configured by the ILSON bit in the RCC_BDCR.
- External 32.768 kHz Oscillator (ELS): Configured by the ELSON bit in the RCC_BDCR.

Exiting the Shutdown Mode

The Shutdown mode is exited according to Exiting the Low-Power Mode. A power-on reset occurs when exiting the shutdown mode. All registers (except those in the backup domain) are reset after wakeup from shutdown. Refer to Table 4-7 for more details on how to exit shutdown mode.

Table 4-7 Shutdown Mode

Standby1 Mode	Description
Mode Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: • SLEEPDEEP bit is set in CMStar with FPU system control register • No interrupt (for WFI) or event (for WFE) is pending • LPMS = “0100” in PMU_CR01 • WUFx bits are cleared in power status register 1 (PMU_SR1)
	On return from ISR while: • SLEEPDEEP bit is set in CMStar with FPU system control register • SLEEPONEXIT = 1 • No interrupt is pending • LPMS = “0100” in PMU_CR01 • WUFx bits are cleared in power status register 1 (PMU_SR1) • The RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup or timestamp flags) is cleared.

Standby1 Mode	Description
Mode Exit	WKUPx pin edge, RTC event, external Reset in NRST pin, ELS CSS failure
Wakeup Latency	Reset phase

4.3.3.2.6 Battery Replacement Mode

NOTE: Refer to device specific to check if this mode is supported or not.

The battery replacement mode is supported as required in some applications. RTC calendar keeps working for a short period in this mode. The time duration relies on the external capacitor or battery on V_{BAT} pin.

When the system detects that the main power is interrupted, PMU automatically switches to battery replacement mode. In this mode, all peripherals in the main power domain are powered off. Only the RTC calendar keeps working.

When the main power is recovered, the whole system will be resumed after the POR reset. The RTC Calendar can be reset only by software bit on RTC registers.

4.3.4 Power Domain Mode

Work Mode	Power Domain					
	PD_VRTC_LV/ VDD_RTC	PD_VDD_HV/ VDD	PD_VMIX_LV/ VMIX	PD_VTSR_LV/ VTSR	PD_VCORE_LV/ VDD_CORE	PD_VDDA_HV/ VDDA
Run	On/0.9 V	On/3.3 V	On/1.1 V	On/1.1 V	On/1.1 V	
LPRun	On/0.9 V	On/3.3 V	On/1.0 V	On/1.0 V	On/1.0 V	
Sleep	On/0.9 V	On/3.3 V	On/1.1 V	On/1.1 V	On/1.1 V	
LPSleep	On/0.9 V	On/3.3 V	On/1.0 V	On/1.0 V	On/1.0 V	
Standby1	On/0.9 V	On/3.3 V	On/0.9 V	On/0.9 V	Off	
Standby2	On/0.9 V	On/3.3 V	On/0.9 V	Off	Off	
Shutdown	On/0.9 V	On/3.3 V	On/0.9 V	Off	Off	
Battery Changing	On/0.9 V	Off	On/0.9 V	Off	Off	

4.3.5 Power Mode Transition

Figure 4-3 shows the transition of power modes.

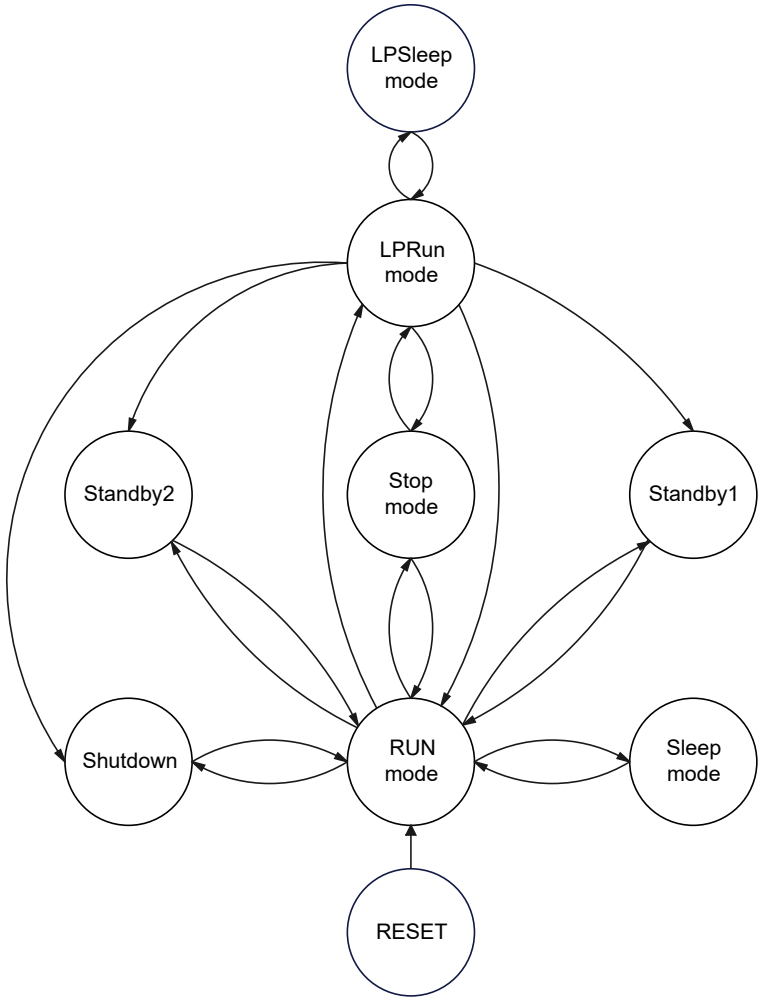


Figure 4-3 Power Mode Transition

	Run	Sleep	LPRun	LPSleep	Stop	Standby1	Standby2	Shutdown
Run	-	WFI/WFE	LPR = 1	--	LPR = 0 LPMS = 4'b1000 + SLEEPDEEP = 1 + WFI/WFE	LPR = 0 LPMS = 4'b0001 + SLEEPDEEP = 1 + WFI/WFE	LPR = 0 LPMS = 4'b0010 + SLEEPDEEP = 1 + WFI/WFE	LPR = 0 LPMS = 4'b0100 + SLEEPDEEP = 1 + WFI/WFE
Sleep	Any interrupt/ wakeup event	-	-	-		-	-	-
LPRun	LPR = 0 Any interrupt/ wakeup event awake to LPRun mode	-	-	LPR = 1, +WFI/WFE	LPR = 1 LPMS = 4'b1000 + SLEEPDEEP = 1 + WFI/WFE	LPR = 1 LPMS = 4'b0001 + SLEEPDEEP = 1 + WFI/WFE	LPR = 1 LPMS = 4'b0010 + SLEEPDEEP = 1 + WFI/WFE	LPR=1 LPMS = 4'b0100 + SLEEPDEEP = 1 + WFI/WFE
LPSleep		-		-		-	-	-
Stop	EXTI, IWDG, NRST		EXTI, IWDG, NRST					
Standby1	LPR = 0: BOR, NRST, RTC, IWDG, GPIO with wakeup cell, SVS	-		-		-	-	-
Standby2	BOR, NRST, RTC, IWDG, GPIO with wakeup cell, SVS	-	-	-		-	-	-
Shutdown	NRST, RTC, GPIO with wakeup cell	-	-	-		-	-	-

4.4 Registers

4.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	PMU_CR00	Power control register 0 This register is reset after wakeup from standby mode.
0x0004	PMU_CR01	Power control register 1 This register is reset after wakeup from standby mode.
0x000c	PMU_CR03	Power control register 3 This register is not reset when exiting standby modes and with the PMURST bit in the RCC_APB1RSTR1 register. Access: Additional APB cycles are needed to access this register versus a standard APB access
0x0014	PMU_CR05	Power control register 5 This register is not reset when exiting standby modes and with the PMURST bit in the RCC_APB1RSTR1 register. Access: Additional APB cycles are needed to access this register versus a standard APB access
0x00a8	PMU_CR2A	Power control register 2A This register is not reset when exiting Standby modes and with the PMURST bit in the RCC_APB1RSTR1 register. Access: Additional APB cycles are needed to access this register versus a standard APB access
0x00ac	PMU_CR2B	Power control register 2B This register is not reset when exiting Standby modes and with the PMURST bit in the RCC_APB1RSTR1 register.

Offset	Register Name	Register Description
		Access: Additional APB cycles are needed to access this register versus a standard APB access
0x00b0	PMU_SR1	Power status register 1 This register is not reset when exiting Standby modes and with the PMURST bit in the RCC_APB1RSTR1 register. Access: Additional APB cycles are needed to access this register versus a standard APB access
0x00b4	PMU_CSR1	Power status clear register Access: Additional APB cycles are needed to access this register versus a standard APB access
0x00b8	PMU_SR2	Power status register 2 This register is partially reset when exiting standby/shutdown modes.
0x00c8	PMU_CSR4	Power status clear register 4 Access: Additional APB cycles are needed to access this register versus a standard APB access

4.4.2 Register Field Details

4.4.2.1 PMU_CR00

0x0000			Power control register 0											PMU_CR00		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							DBP	Reserved			LPR	Reserved			
Type	RO							RW	RO			RW	RO			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-8 Power Control Register 0 Description

Field	Name	Description
31:9	Reserved	Reserved
8	DBP	Disable backup domain write protection In the 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
7:5	Reserved	Reserved
4	LPR	Low-power run When this bit is set, the LDO_CORE runs in the low-power mode (LPR).

Field	Name	Description
3:0	Reserved	Reserved

4.4.2.2 PMU_CR01

0x0004	Power control register 1													PMU_CR01		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					SRAMSYNC EN	STOPSRAM 2RET	STOPSRAM 1RET1	STOPSRAM 1RET0	STOPSRAM 0RET2	STOPSRAM 0RET1	STOPSRAM 0RET0	LPMS			
Type	RO					RW	RW	RW	RW	RW	RW	RW	RW			
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 4-9 Power Control Register 1 Description

Field	Name	Description
31:11	Reserved	Reserved
10	SRAMSYNCEN	Enable SRAM parameter synchronizer
9	STOPSRAM2RET	When stop the 16 KB SRAM in SRAM2 retention enable: 0: Power down (not support) 1: Retention
8	STOPSRAM1RET1	When stop the second 64 KB SRAM in SRAM1 retention enable: 0: Power down (not support) 1: Retention
7	STOPSRAM1RET0	When stop the first 64 KB SRAM in SRAM1 retention enable: 0: Power down (not support) 1: Retention

Field	Name	Description
6	STOPSRAM0RET2	When stop the third 64 KB SRAM in SRAM0 retention enable: 0: Power down (not support) 1: Retention
5	STOPSRAM0RET1	When stop the second 64 KB SRAM in SRAM0 retention enable: 0: Power down (not support) 1: Retention
4	STOPSRAM0RET0	When stop the first 64 KB SRAM in SRAM0 retention enable: 0: Power down (not support) 1: Retention
3:0	LPMS	Low-power mode selection These bits select the low-power mode to be entered when the CPU enters the deep sleep mode. 0001: Standby1 mode 0010: Standby2 mode 0100: Shutdown mode 1000: Stop mode 0000: Standard mode Others: Should be forbidden **NOTE:** In stop and standby0 mode, SRAM16K is retained.

4.4.2.3 PMU_CR03

0x000c			Power control register 3											PMU_CR03		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															SVSEN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-10 Power Control Register 1 Description

Field	Name	Description
31:1	Reserved	Reserved
0	SVSEN	Programmable voltage detector enable 0: Programmable voltage detector disable. 1: Programmable voltage detector enable.

4.4.2.4 PMU_CR05

0x0014			Power control register 5											PMU_CR05		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	IHSCALUPDENFRC	IHSTRIM													
Type	RO	RW	RW													
Reset	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0

Table 4-11 Power Control Register 5 Description

Field	Name	Description
31:15	Reserved	Reserved
14	IHSCALUPDENFRC	Set 1 before calibrating IHS using IHSTRIM, or keep it at 0.
13: 0	IHSTRIM	IHS trim bits, only affect when IHSCALUPDENFRC = 1.

4.4.2.5 PMU_CR2A

0x00a8			Power control register 2A											PMU_CR2A		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EIWUL	Reserved										EWUP5	EWUP4	EWUP3	EWUP2	EWUP1
Type	RW	RO										RW	RW	RW	RW	RW
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-12 Power Control Register 2A Description

Field	Name	Description
31:16	Reserved	Reserved
15	EIWUL	Enable internal wakeup line 0: Internal wakeup line disabled 1: Internal wakeup line enabled
14:5	Reserved	Reserved
4	EWUP5	Enable wakeup pin WKUP5 When this bit is set, the external wakeup pin WKUP5 is enabled and triggers a wakeup from standby or shutdown event when a rising or falling edge occurs. The active edge is configured via the WP5 bit in the PMU_CR2B register.
3	EWUP4	Enable wakeup pin WKUP4

Field	Name	Description
		When this bit is set, the external wakeup pin WKUP4 is enabled and triggers a wakeup from standby or shutdown event when a rising or falling edge occurs. The active edge is configured via the WP4 bit in the PMU_CR2B register.
2	EWUP3	Enable wakeup pin WKUP3 When this bit is set, The external wakeup pin WKUP3 is enabled and triggers a wakeup from standby or shutdown event when a rising or falling edge occurs. The active edge is configured via the WP3 bit in the PMU_CR2B register.
1	EWUP2	Enable wakeup pin WKUP2 When this bit is set, the external wakeup pin WKUP2 is enabled and triggers a wakeup from standby or shutdown event when a rising or falling edge occurs. The active edge is configured via the WP2 bit in the PMU_CR2B register.
0	EWUP1	Enable wakeup pin WKUP1 When this bit is set, the external wakeup pin WKUP1 is enabled and triggers a wakeup from standby or shutdown event when a rising or falling edge occurs. The active edge is configured via the WP1 bit in the PMU_CR2B register.

4.4.2.6 PMU_CR2B

0x00ac			Power control register 2B											PMU_CR2B		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved											WP5	WP4	WP3	WP2	WP1
Type	RO											RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-13 Power Control Register 2B Description

Field	Name	Description
31:5	Reserved	Reserved
4	WP5	Wakeup pin WKUP5 polarity This bit defines the polarity used for event detection on the external wake-up pin, WKUP5. 0: Detection on high level (rising edge) 1: Detection on low level (falling edge)
3	WP4	Wakeup pin WKUP4 polarity This bit defines the polarity used for event detection on the external wake-up pin, WKUP4. 0: Detection on high level (rising edge) 1: Detection on low level (falling edge)
2	WP3	Wakeup pin WKUP3 polarity This bit defines the polarity used for event detection on the external wake-up pin, WKUP3. 0: Detection on high level (rising edge)

Field	Name	Description
		1: Detection on low level (falling edge)
1	WP2	Wakeup pin WKUP2 polarity This bit defines the polarity used for event detection on the external wake-up pin, WKUP2. 0: Detection on high level (rising edge) 1: Detection on low level (falling edge)
0	WP1	Wakeup pin WKUP1 polarity This bit defines the polarity used for event detection on the external wake-up pin, WKUP1. 0: Detection on high level (rising edge) 1: Detection on low level (falling edge)

4.4.2.7 PMU_SR1

0x00b0			Power status register 1											PMU_SR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	WUFI	Reserved				SDF	SB2F	SB1F	Reserved			WUF5	WUF4	WUF3	WUF2	WUF1
Type	RO	RO				RO	RO	RO	RO			RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-14 Power Status Register 1 Description

Field	Name	Description
31:16	Reserved	Reserved
15	WUFI	Wakeup flag internal This bit is set when a wakeup is detected on the internal wakeup line. It is cleared when all internal wakeup sources are cleared.
14:11	Reserved	Reserved
10	SDF	Shutdown flag This bit is set by hardware when the device enters the shutdown mode and is cleared by setting the CSBF bit in the PMU_SCR register or by a power-on reset. It is not cleared by the system reset. 0: The device did not enter the standby mode 1: The device entered the standby mode
9	SB2F	Standby2 flag

Field	Name	Description
		This bit is set by hardware when the device enters the standby2 mode and is cleared by setting the CSBF bit in the PMU_SCR register or by a power-on reset. It is not cleared by the system reset. 0: The device did not enter the standby mode 1: The device entered the standby mode
8	SB1F	Standby1 flag This bit is set by hardware when the device enters the standby1 mode and is cleared by setting the CSBF bit in the PMU_SCR register or by a power-on reset. It is not cleared by the system reset. 0: The device did not enter the standby mode 1: The device entered the standby mode
7:5	Reserved	Reserved
4	WUF5	Wakeup flag 5 This bit is set when a wakeup event is detected on the wakeup pin, WKUP5. It is cleared by writing '1' in the CWUF5 bit of the PMU_SCR register.
3	WUF4	Wakeup flag 4 This bit is set when a wakeup event is detected on the wakeup pin, WKUP4. It is cleared by writing '1' in the CWUF4 bit of the PMU_SCR register.
2	WUF3	Wakeup flag 3 This bit is set when a wakeup event is detected on the wakeup pin, WKUP3. It is cleared by writing '1' in the CWUF3 bit of the PMU_SCR register.
1	WUF2	Wakeup flag 2 This bit is set when a wakeup event is detected on the wakeup pin, WKUP2. It is cleared by writing '1' in the CWUF2 bit of the PMU_SCR register.
0	WUF1	Wakeup flag 1

Field	Name	Description
		This bit is set when a wakeup event is detected on the wakeup pin, WKUP1. It is cleared by writing '1' in the CWUF1 bit of the PMU_SCR register.

4.4.2.8 PMU_CSR1

0x00b4			Power status clear register											PMU_CSR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							CSBF	Reserved			CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
Type	RO							RW	RO			RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-15 Power Status Clear Register Description

Field	Name	Description
31:9	Reserved	Reserved
8	CSBF	Clear standby flag Setting this bit clears the SBF flag in the PMU_SR1 register.
7:5	Reserved	Reserved
4	CWUF5	Clear wakeup flag 5 Setting this bit clears the WUF5 flag in the PMU_SR1 register.
3	CWUF4	Clear wakeup flag 4 Setting this bit clears the WUF4 flag in the PMU_SR1 register.
2	CWUF3	Clear wakeup flag 3 Setting this bit clears the WUF3 flag in the PMU_SR1 register.
1	CWUF2	Clear wakeup flag 2

Field	Name	Description
		Setting this bit clears the WUF2 flag in the PMU_SR1 register.
0	CWUF1	Clear wakeup flag 1 Setting this bit clears the WUF1 flag in the PMU_SR1 register.

4.4.2.9 PMU_SR2

0x00b8	Power status register 2													PMU_SR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved											SVSOUT1	SVSOUT0	LPF	Reserved	FLASHRDY
Type	RO											RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-16 Power Status Register 2 Description

Field	Name	Description
31:5	Reserved	Reserved
4	SVSOUT1	Programmable voltage detector1 output 0: VDD is above the selected PVD threshold 1: VDD is below the selected PVD threshold
3	SVSOUT0	Programmable voltage detector0 output 0: VDD is above the selected PVD threshold 1: VDD is below the selected PVD threshold
2	LPF	Low-power regulator adjustment flag This bit is set by hardware when the MCU enters or exits low-power run mode. When the MCU enters or exits the low-power run mode, this bit remains 1 until the regulator is ready. It is recommended to poll this bit before increasing the CPU frequency. This bit is cleared by hardware when the regulator is ready.

Field	Name	Description
		0: The regulator adjustment is completed 1: The regulator adjustment is in progress
1	Reserved	Reserved
0	FLASHRDY	Flash ready flag This bit is set by hardware to indicate the Flash memory is ready to be accessed after wakeup from power-down. To power down the Flash memory, set the FPD_LPRUN, FPD_LPSLP, or FPD_STP bits. 0: Flash memory power down 1: Flash memory is ready to be accessed **NOTE:** When the system boots from SRAM, the user needs to wait for the FLASHRDY bit to be set before jumping to Flash memory.

4.4.2.10 PMU_CSR4

0x00c8			Power status clear register 4											PMU_CSR4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		IHSTRIM													
Type	RO		RO													
Reset	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0

Table 4-17 Power Status Clear Register 4 Description

Field	Name	Description
31:14	Reserved	Reserved
13: 0	IHSTRIM	The IHS trim value in use

Reset and Clock Control (RCC)

This chapter describes the details of the Reset and Clock Control (RCC).

Topics: **Page**

5.1 Reset and Clock Control (RCC) Pinout

Signal Name	Signal Type	Pin Description
NRST	I/O	System reset
OSC_IN	I	External high speed oscillator input
OSC_OUT	O	External high speed oscillator output
OSC32_IN	I	External low speed oscillator input
OSC32_OUT	O	External low speed oscillator output
MCO	O	Clock output for external device
LSCO	O	Low speed clock output for external device

5.2 Reset Introduction

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

5.2.1 Power Reset

A power reset is triggered when a Power-on Reset (POR) or a Brown-out Reset (BOR) event occurs.

A BOR, including Power-on or Power-down Reset (POR/PDR), sets all registers to their reset values except the RTC domain.

When exiting standby or shutdown mode, all registers in the V_{CORE} domain are set to their reset value. Registers outside the V_{CORE} domain (RTC, WKUP, IWDG, and standby/shutdown modes control) are not impacted.

5.2.2 System Reset

A system reset sets all registers to their reset values except the reset flags in the RCC clock control/ status register (RCC_CSR) and the registers in the RTC domain. A system reset is generated when one of the following events occurs:

- A low level on the NRST pin (external reset)
- Window Watchdog event (WWDG reset)
- Independent Watchdog event (IWDG reset)
- A software reset (SW reset) (refer to Software Reset)
- Low-power mode security reset (refer to Low-Power Mode Security Reset)
- Option byte loader reset (refer to Option Byte Loader Reset)
- A Brown-out Reset (BOR)

The reset source can be identified by checking the reset flags in the RCC_CSR.

5.2.2.1 NRST Pin (External Reset)

Through specific option bits (NRSTMODE), the NRST pin is configurable for operating as:

- **Reset input/output**

Any valid reset signal on the pin is propagated to device internal logic and all internal reset sources are externally driven through a pulse generator to this pin. The pulse generator guarantees a minimum reset pulse duration of 20 μs for each internal reset source to be output on the NRST pin. In case on an internal reset, the internal pull-up RPU is deactivated to save the power consumption

through the pull-up resistor. This mode is always active (independently of the option bytes setting) during each device power-on-reset (until option bytes are loaded): power on the device.

- **Reset input (default at device delivery)**

In this mode, any valid reset signal on the NRST pin is propagated to device internal logic, but resets generated internally by the device are not visible on the pin.

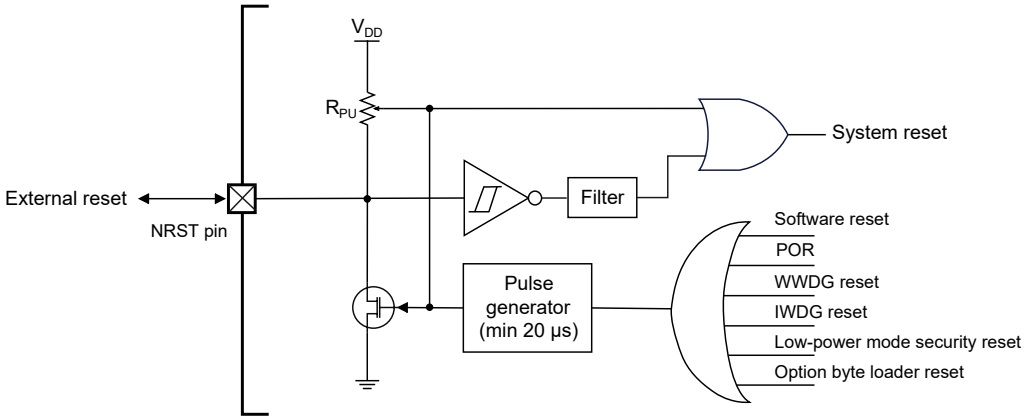


Figure 5-1 Simplified Diagram of Reset Circuit

5.2.2.2 Software Reset

The SYSRESETREQ bit in ARM China STAR processor Application Interrupt and Reset Control (AIRCR) register must be set to force a software reset on the device.

5.2.2.3 Low-Power Mode Security Reset

To prevent that critical applications mistakenly enter a low-power mode, three low-power mode security resets are available. If enabled in option bytes, the resets are generated in the following conditions:

- **Entering standby mode**:

  This type of reset is enabled by setting NRSTSTDBY bit in user option bytes. In this case, whenever a standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
- **Entering stop mode** :

  This type of reset is enabled by setting NRSTSTOP bit in user option bytes. In this case, whenever a stop mode entry sequence is successfully executed, the device is reset instead of entering stop mode.
- **Entering shutdown mode**:

  This type of reset is enabled by setting NRSTSHDW bit in user option bytes. In this case, whenever a shutdown mode entry sequence is successfully executed, the device is reset instead of entering shutdown mode.

For further information on the user option bytes, refer to Option Byte Loader Reset.

5.2.2.4 Option Byte Loader Reset

The option byte loader reset is generated when the OBLLAUNCH bit is set in the FMC register. This bit is used to launch the option byte loading by software.

5.2.3 RTC Domain Reset

The RTC domain has two specific resets.

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

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

A RTC domain reset affects the ELS and ILS oscillator and the entire RTC domain registers.

5.3 Clock Introduction

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

- 8 MHz high-speed internal RC oscillator (IHS)
- 4 ~ 32 MHz high-speed oscillator with external crystal or ceramic resonator (EHS)
- PLL clock

The IHS is used as system clock source after startup from reset.

The devices have the following additional clock sources:

- 32768 Hz low-speed oscillator with external crystal (ELS), supporting eight drive capability modes
- 32768 Hz low-speed internal RC oscillator (ILS) with 2% accuracy

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

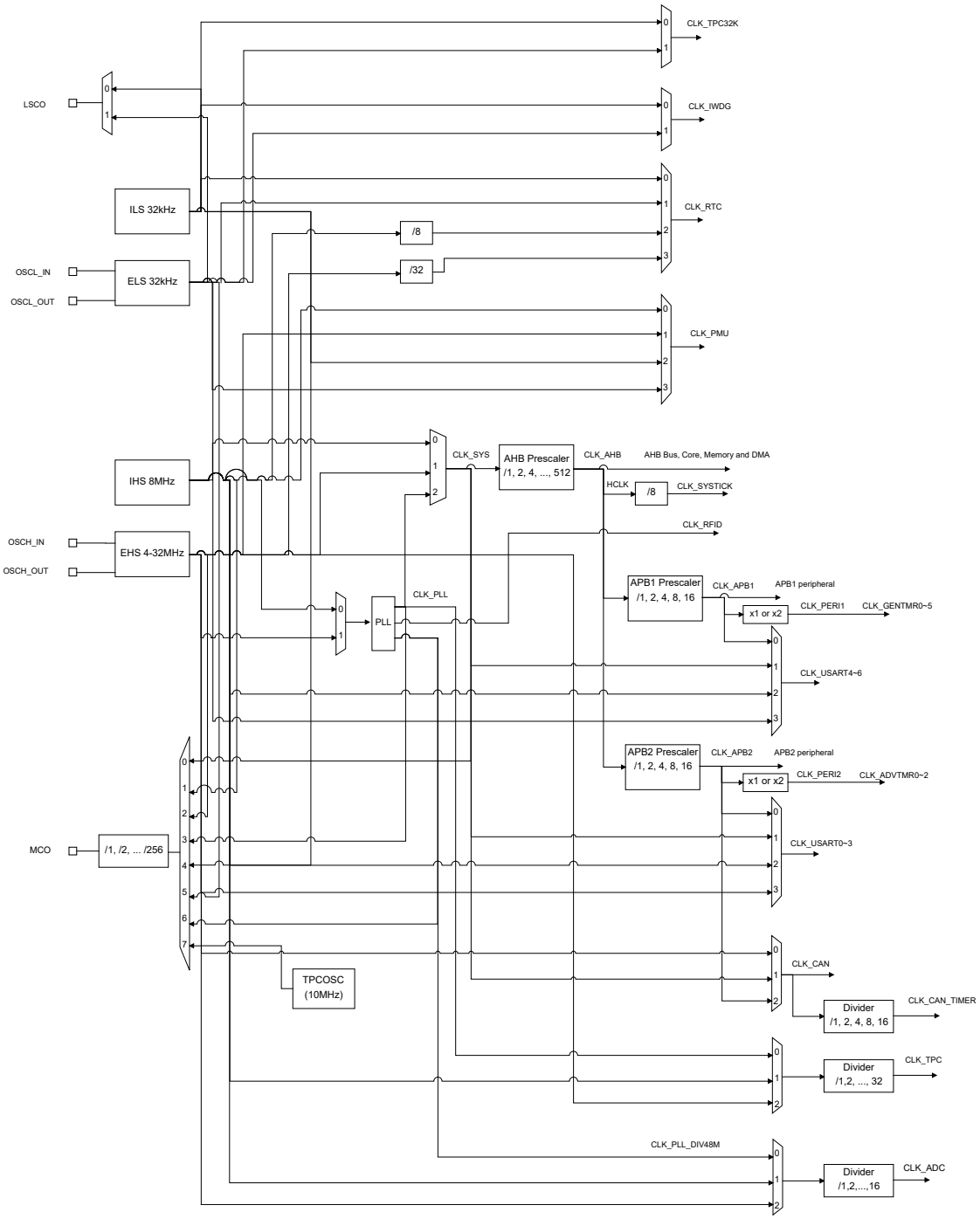


Figure 5-2 RCC Clock Tree

PLL block has three outputs:

- System clock (up to 156 MHz)
- ADC clock (PLL VCO outputs 360 MHz, divide by 4 by default and sources 90 MHz for ADC)
- RFID clock (13.56 MHz)

Several prescalers can be used to configure the AHB frequency, the APB1 and APB2 domains. The maximum frequency of the AHB, the APB1 and the APB2 domains is 156 MHz.

All the peripheral clocks are derived from their bus clock CLK_AHB, CLK_PERI1~2, CLK_APB1~2) except:

- The PMU clock is derived from one of the following sources:

  - IHS clock
  - EHS clock (not supported)
  - ILS clock
  - ELS clock
- The TPSENSOR clock is derived from one of the following sources:
  - Main PLL clock from CLK_PLL
  - IHS clock
  - EHS clock
- The ADC clock is derived from one of the following sources:
  - Additional PLL clock from CLK_PLL_DIV48M
  - IHS clock
  - EHS clock
- The CAN clock is derived from one of the following sources:
  - EHS clock
  - System clock
  - Peripheral clock from CLK_APB2
- The USART0~3 clocks are derived from one of the following sources:
  - Peripheral clock from CLK_APB2
  - System clock
  - IHS clock
  - ELS clock
- The USART4~6 clock are derived from one the following sources:
  - Peripheral clock from CLK_APB1
  - System clock
  - IHS clock
  - ELS clock
- The RTC clock is derived from one of the following sources:
  - ELS clock
  - ILS clock
  - EHS clock divide by 32
  - IHS clock divide by 8
- The IWDG block is derived from one of the following sources:
  - ILS clock
  - ELS clock

The RCC feeds the Cortex® System Timer (SysTick) external clock with the AHB clock (CLK_AHB) divided by 8. The SysTick can work either with this clock or directly with the Cortex® clock, configurable in the SysTick Control and Status Register.

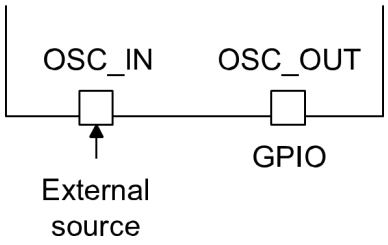
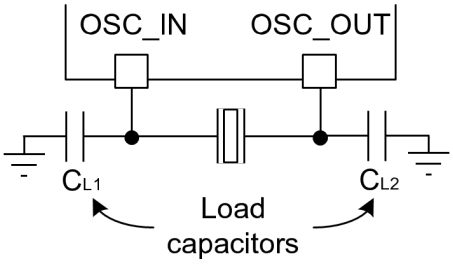
5.3.1 External High-Speed (EHS) Clock

The External High-Speed clock signal (EHS) can be generated from two possible clock sources:

- External crystal/ceramic resonator
- External clock

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

Table 5-1 EHS/ELS Clock Sources

Clock Source	Hardware Configuration
External clock	
Crystal/Ceramic resonators	

External Source (EHS Bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 32 MHz. You select this mode by setting the EHSBYP and EHSON bits in the clock control register (RCC_CR). The external clock signal (square, sine or triangle) with 50% duty cycle depending on the frequency (refer to the datasheet) has to drive the OSCH_IN pin while the OSCH_OUT pin can be used as GPIO. Refer to Table 5-1.

External Crystal/Ceramic Resonator (EHS Crystal)

The 4 to 32 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in Table 5-1. Refer to the electrical characteristics section of the datasheet for more details.

The EHSRDY flag in the clock control register (RCC_CR) indicates if the EHS 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 enable register (RCC_CIER).

The EHS Crystal can be switched on and off using the EHSON bit in the clock control register (RCC_CR). But if any of the following conditions meet, EHS cannot be disabled:

- EHS clock used as source for system clock
- When PLL is enabled and used as source for system clock, and EHS is used as PLL reference clock

For above conditions, EHSON cannot be set to 0, even firmware set this bit as 0, EHSRDY remains 1, and EHS is not disabled.

NOTE: When the MCU enters stop, standby or shutdown mode, the EHS is automatically disabled by the hardware. If EHS/32 is used as the source for the RTC clock, the RTC will also be stopped.

5.3.2 External Low-Speed (ELS) Clock

The External Low-Speed (ELS) crystal is a 32768 Hz ELS crystal or ceramic resonator. It has the advantage of providing a low-power but highly accurate clock source to the Real-Time Clock peripheral (RTC) for clock/calendar or other timing functions.

The ELS crystal is switched on and off using the ELSON bit in the RTC domain control (RCC_BDCR) register. The crystal oscillator driving strength can be changed at runtime using the ELSCTRL[7:0] bits in the RCC_BDCR register to obtain the best compromise between robustness and short startup time on one side and low-power consumption on the other side.

The ELSRDY flag in the RCC_BDCR register indicates whether the ELS crystal is stable or not. At startup, the ELS crystal output clock signal is not released until this bit is set by the hardware. An interrupt can be generated if enabled in the clock interrupt enable register (RCC_CIER).

When the MCU enters stop or standby mode, ELS remains enabled. ELS will not be disabled with any of the following conditions:

- ELS clock used as a source for PMU
- ELS clock used as a source for IWDG

ELS clock can be driven on either MCO or LSCO pin.

External Source (ELS Bypass)

In this mode, an external clock source must be provided. Select this mode by setting the ELSBYP and ELSON bits in the RCC_BDCR register. The external clock signal (square, sine or triangle) with ~50% duty cycle has to drive the OSCL_IN pin while the OSCL_OUT pin can be used as GPIO. Refer to Table 5-1.

5.3.3 Internal High-Speed (IHS) Clock

The Internal High-Speed (IHS) clock signal is generated from an internal 8 MHz RC Oscillator, it can be used as source for system clock, some peripheral clock and PLL.

The IHS RC oscillator has the following advantages:

- Provide a clock source at low cost (no external components)
- Faster startup time than the EHS crystal oscillator

However, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

The IHSRDY flag in the clock control register (RCC_CR) indicates if the IHS oscillator is stable or not. At startup, the IHS output clock is not released until this bit is set by hardware.

IHS can be enabled and disabled with IHSON bit, if any of the following conditions are met, IHS cannot be disabled:

- IHS clock used as source for system clock
- When PLL is enabled and used as source for system clock, and IHS is used as PLL reference clock
- IHS clock used as source for PMU clock

The IHS clock can be selected as system clock after wakeup from stop mode. It can also be used as a backup clock source (auxiliary clock) if the EHS crystal oscillator fails. Refer to Clock Security System (CSS).

NOTE: When entering the stop, standby or shutdown mode, the IHS clock is disabled. If the IHS/8 is used for RTC clock, RTC will be stopped.

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations.

After reset, the factory calibration value is loaded in the IHSTRIM[13:0] bits in the PMU_CSR4 register.

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. You can trim the IHS frequency in the application using the IHSTRIM[13:0] in the PMUCR05 register. User need to first set IHSCALUPDENFRC bit in the PMUCR05 to 1, then configure IHSTRIM[13:0] in this register.

For more details on how to measure the IHS frequency variation, refer to Internal/external clock measurement with GPTMR0~1.

5.3.4 Internal Low-Speed (ILS) Clock

The ILS RC acts as a low-power clock source that can be kept running in stop and standby mode for the Independent Watchdog (IWDG) and RTC. The clock frequency is 32768 Hz. For more details, refer to the electrical characteristics section of the datasheets.

The ILS RC can be switched on and off using the ILSON bit in the RTC domain control register (RCC_BDCR). If IWDG is enabled and choosing ILS as clock source or PMU chooses ILS as clock source, then ILS will be enabled by default and cannot be disabled.

The ILSRDY flag in the RTC domain control register (RCC_BDCR) indicates if the ILS 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 enable register (RCC_CIER).

5.3.5 Phase Lock Loop (PLL)

The internal Phase Lock Loop (PLL) can be used to multiply the IHS or EHS output clock frequency. It provides three outputs:

- Main PLL output (CLK_PLL) for supplying system clock
- RFID clock (CLK_RFID)
- Additional PLL output (CLK_PLL_DIV48M) to provide independent clock for ADC

PLL Configuration

The PLL input frequency must be within the range defined in the device datasheet. The selected clock source is divided by a programmable factor PLLINDIV in RCC_PLLCFGR1 register to provide a clock frequency in the requested input range. In addition, PLLLPFC1/C2/C3 bits in RCC_PLLCFGR3 register need to be set based on two different input clock ranges, 4 ~ 8 MHz or 8 ~ 32 MHz as shown below table.

	PLL Input Clock in 4 ~ 8 MHz	PLL Input Clock in 8 ~ 32 MHz
RCC_PLLCFGR3[PLLLPFC1]	0x14	0x02
RCC_PLLCFGR3[PLLLPFC2]	0x0A	0x08

RCC_PLLCFGR3[PLLLPFC3]	0x14	0x04

NOTE: By default, PLLLPFC1/C2/C3 bits is set to support PLL input clock in range of 8 ~ 32 MHz.

The PLL configuration (selection of the input clock and multiplication factor) must be done before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLON to 0 in clock control register (RCC_CR).
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Choose PLL source clock with PLLSRC in RCC_PLLCFGR1 register
4. Change the desired parameter based on which PLL output need to be configured, refer to RCC_PLLCFGR1 and RCC_PLLCFGR2 registers for details.
   a. PLLFBDIV is to configure the multiplication factor for PLL input clock to get PLL VCO.
   b. PLLODIV is to set output divider for main PLL output (CLK_PLL).
   c. PLLRDIVINT and PLLRDIVFRAC is to set integer and fractional divider for RFID clock (CLK_RFID).
   d. PLLDIV48M is to set output divider for the additional PLL output (CLK_PLL_DIV48M) for ADC.
5. Enable the PLL again by setting PLLON to 1.
6. Enable additional PLL outputs by clearing bits in RCC_PLLCFGR1 or RCC_PLLCFGR2 register.
   a. If need to output CLK_PLL_DIV48M for ADC block, then PLLDIV48MPD need to be cleared to 0.
   b. If need to output CLK_RFID, then PLLRDIVPD need to be cleared to 0.

Refer to the following equations for how to set PLL parameters to generate different PLL outputs based on PLL input clock (Fin).

PLL Outputs	PLL Clock Equations
Main PLL (CLK_PLL)	$F_{clk\_pll} = (F_{in} / PLLINDIV) \times PLLFBDIV / PLLODIV$
RFID clock (CLK_RFID)	$F_{clk\_rfid} = (F_{in} / PLLINDIV) \times PLLFBDIV / (PLLRDIVINT + PLLRDIVFRAC \times 0.25) / 2$
Additional PLL output for ADC (CLK_PLL_DIV48M)	$F_{clk\_pll\_div48m} = (F_{in} / PLLINDIV) \times PLLFBDIV / PLLDIV48M$

An interrupt can be generated when the PLL is ready, if enabled in the clock interrupt enable register (RCC_CIER).

The PLL output frequency must not exceed 156 MHz.

5.3.6 System Clock Selection

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

- IHS clock
- EHS clock
- PLL

The system clock maximum frequency is 156 MHz. After a system reset, the IHS oscillator is selected as system clock. When a clock source is used directly or through the PLL as a 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 occurs when the clock source becomes ready.

5.3.7 Clock Security System (CSS)

The Clock Security System (CSS) can be activated by enabling the CSSON bit. In this case, the clock detector is enabled after the EHS oscillator startup delay and disabled when this oscillator is stopped. When the MCU resets or enters standby mode, the CSSON bit is cleared by hardware.

If a failure is detected on the EHS clock, and if the EHS oscillator is used directly or indirectly as the system clock (indirectly means it is used as the PLL input clock and the PLL clock is used as the system clock), then the system clock will automatically switch to IHS to provide a secure clock, then EHS oscillator is automatically disabled.

A clock failure event is sent to the break input of the General-Purpose Timer (GPTMR0~5) or Advanced-Control Timers (ADVTMR0~2), and an interrupt is generated to inform the software about the failure (Clock Security System Interrupt, CSSI), allowing the MCU to perform rescue operations by first switch system clock to IHS and then disable EHS and CSS. If EHS provides a source clock for PLL when clock failure occurs, PLL will be disabled as well. The CSSI is linked to the Cortex Star Core with an FPU Non-maskable Interrupt (NMI) exception vector.

NOTE: Once the CSS is enabled and if the EHS clock fails, the CSS interrupt occurs, and an NMI is automatically generated. RCC clock interrupt flag register (RCC_CIFR) will set the CSSF flag to indicate the failure. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. Consequently, in the NMI ISR, the user must clear the CSS interrupt by setting the CSSC bit in the clock interrupt clear register (RCC_CICR).

CSS on ELS

A CSS on ELS can be activated by software writing the ELSCSSON bit in the RCC control/status register (RCC_BDCR). This bit can be disabled only by a hardware reset or RTC software reset or after a failure detection on ELS. ELSCSSON must be written after ELS is enabled (ELSON set to ‘1’) and ready (ELSRDY set by hardware) and after the RTC clock has been selected by RTCSEL.

The CSS on ELS works in all modes except VBAT (battery changing mode). If a failure is detected on the external 32 kHz oscillator, the ELS clock is no longer supplied to the RTC, but RTCSEL, ELSCSSON and ELSON bit are not changed by hardware. ELS clock will be disabled as well.

In standby and shutdown modes, a wakeup is generated. In other modes, an interrupt can be sent to wake up the software. This interrupt is mapped to RTC/CSS_ELS through EXTI-18. The software then must disable the ELSCSSON bit, stop the failure 32 kHz oscillator (clearing ELSON), and change the RTC clock source (ILS, EHS/32 or IHS/8) with RTCSEL bit, or take any required action to secure the application.

5.3.8 ADC Clock

The ADC clock is derived from the CLK_PLL_DIV48M, IHS or EHS clock. It can be divided first by ADCDIV[3:0] in RCC_CFG2R register, then by the PRESC[3:0] in the ADC_CCR register. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bits in the ADC_CCR register.

Refer to the device datasheet for the maximum ADC clock.

5.3.9 RTC Clock

The RTCCLK clock source can be either the ELS, ILS, EHS/32 or IHS/8. It is selected by programming the RTC_SEL[1:0] bits in the RTC domain control register (RCC_BDCR). This selection cannot be modified without resetting the RTC domain.

The ELS and ILS clocks are in the RTC domain, whereas the EHS and IHS clocks are not. Consequently:

If ELS or ILS 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 EHS/32 or IHS/8 is selected as the RTC clock:

- The RTC state is not guaranteed if the VDD supply is powered off.

5.3.10 Clock-out Capability

MCO

The Microcontroller Clock Output (MCO) capability allows the clock to be output onto the external MCO pin. One of eight clock signals can be selected as the MCO clock.

- System clock
- IHS clock
- EHS clock
- Main PLL clock
- ILS clock
- ELS clock
- ADC PLL clock (CLK_PLL_DIV48M) clock
- TPCOSC (10 MHz) clock (only available under MCU test mode)

The selection is controlled by the MCOSEL[2:0] bits of the clock configuration register (RCC_CFGR). The selected clock can be divided with the MCOPRE[7:0] field of the RCC_CFGR.

LSCO

The Low-Speed Clock Output (LSCO) allows a low-speed clock to be output onto the external LSCO pin:

- ILS
- ELS

This output remains available in stop modes. The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN in the RTC control/status register (RCC_CSR).

The MCO or LSCO clock output requires the corresponding alternate function (AF0) selected on the pins.

5.3.11 Internal/External Clock Measurement with GPTMR0-1

It’s possible to indirectly measure the frequency of all on-board clock sources by means of the GPTMR0~1 channel 0 input capture, as represented in the below figures.

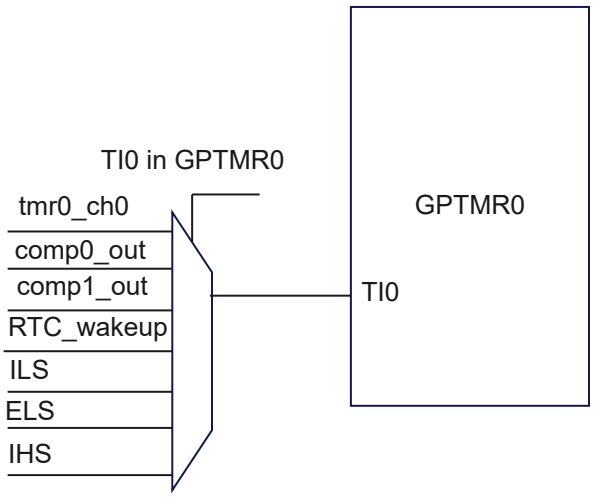


Figure 5-3 Frequency Measurement with GPTMR0 in Capture Mode

The input capture channel of the GPTMR0 can be a GPIO line or an internal clock of the MCU. The possibilities are the following:

GPTMR0 channel0 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.

- GPTMR0 channel0 is connected to comp0_out
- GPTMR0 channel0 is connected to comp1_out
- GPTMR0 channel0 is connected to the RTC wakeup interrupt signal. In the case, the RTC interrupt should be enabled.
- GPTMR0 channel0 is connected to the ILS
- GPTMR0 channel0 is connected to the ELS
- GPTMR0 channel0 is connected to the IHS

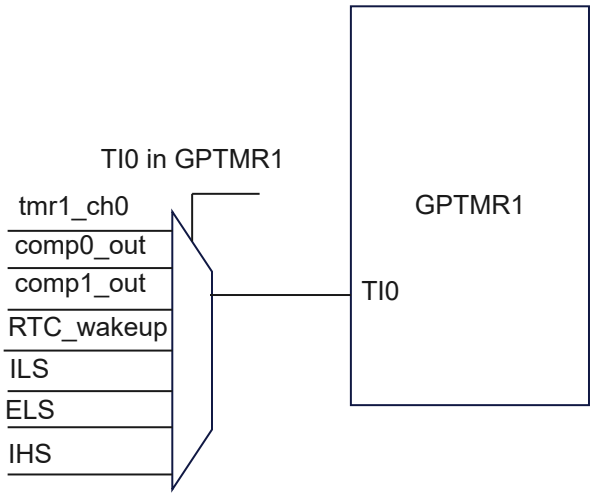


Figure 5-4 Frequency Measurement with GPTMR1 in Capture Mode

The input capture channel of the GPTMR1 can be a GPIO line or an internal clock of the MCU. The possibilities are the following ones:

- GPTMR1 channel0 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- GPTMR1 channel0 is connected to comp0_out
- GPTMR1 channel0 is connected to comp1_out

- GPTMR1 channel0 is connected to the RTC wakeup interrupt signal. In the case the RTC interrupt should be enabled.
- GPTMR1 channel0 is connected to the ILS
- GPTMR1 channel0 is connected to the ELS
- GPTMR1 channel0 is connected to the IHS

5.3.11.1 IHS Calibration

For GPTMR0 and GPTMR1, the primary purpose of connecting the ELS to the channel 0 input capture is to precisely measure the IHS system clocks. To do this, the IHS should be used as the system clock source.

The measurement is based on the number of IHS clock counts between consecutive edges of the ELS signal. This provides a measure of the internal clock period.

Key advantages and features include:

- The high precision of ELS crystals, typically a few tens of parts per million (ppm's), allows for the determination of the internal clock frequency with the same resolution.
- The source can be trimmed to compensate for frequency deviations related to manufacturing, process, temperature, and voltage.
- The IHS oscillator has dedicated user-accessible calibration bits for calibration purposes.

The basic concept of this process is to provide a relative measurement, such as the IHS/ELS ratio. The precision of the measurement is closely related to the ratio between the two clock sources. The higher the ratio, the better the precision of the measurement.

5.3.11.2 ILS Calibration

The calibration process for the ILS follows a similar pattern to that of the IHS, but with a change in the reference clock. It requires the connection of the ILS clock to the channel 0 input capture of either GPTMR0 or GPTMR1.

Next, the EHS should be defined as the system clock source. This allows the number of EHS clock counts between consecutive edges of the ILS signal to provide a measure of the internal low-speed clock period.

The basic concept involves providing a relative measurement, such as the EHS/ILS ratio. The precision of this measurement is closely tied to the ratio between the two clock sources. A higher ratio results in a more accurate measurement.

5.3.12 Peripheral Clock Enable Register (RCC_AHBxENR, RCC_APBxENR)

Each peripheral clock can be enabled by the xxxxEN bit of the RCC_AHBxENR, RCC_APBxENRy registers.

When the peripheral clock is not enabled, the peripheral registers read or write accesses are not supported and will trigger a hard fault exception.

The enable bit has a synchronization mechanism to create a glitch free clock for the peripheral. After the enable bit is set, there is a 2 clock cycles delay before the clock be active.

CAUTION

Just after enabling the clock for a peripheral, software must wait for a delay before accessing the peripheral registers.

5.4 Low-Power Modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and low-power sleep modes stops the CPU clock. The memory interface clocks (Flash and SRAM interfaces) can be stopped by software during sleep mode.
- Stop, standby and shutdown mode stops all the clocks in the V_{CORE} domain and disables the PLL, IHS and EHS oscillator.

The CPU's deepsleep mode can be overridden for debugging by setting the DBGSTOP, DBGSTANDBY or DBGSHUTDOWN bits in the DBGMCU_CR register.

When exiting the stop modes (stop, standby or shutdown), the system clock is IHS.

If a Flash memory programming operation is ongoing, the entry into Stop, standby and shutdown modes is delayed until the Flash memory interface access is completed. Similarly, if an access to the APB domain is ongoing, the entry into stop, standby and shutdown modes entry is delayed until the APB access is finished.

5.5 Registers

5.5.1 Register Address Map

Offset	Register Name	Register Description
0x0000	RCC_CR	Clock control register
0x0004	RCC_CR1	Clock control register 1
0x0008	RCC_CFGR	Clock configuration register
0x000c	RCC_PLLCFGR1	PLL configuration register
0x0010	RCC_PLLCFGR2	PLL configuration register 2
0x0014	RCC_CR2	Clock control register 2
0x0018	RCC_CIER	Clock interrupt enable register
0x001c	RCC_CIFR	Clock interrupt flag register
0x0020	RCC_CICR	Clock interrupt clear register
0x0028	RCC_AHB1RSTR	AHB1 peripheral reset register
0x002c	RCC_AHB2RSTR	AHB2 peripheral reset register
0x0030	RCC_AHB2RSTR1	AHB2 peripheral reset register 1
0x0038	RCC_APB1RSTR	APB1 peripheral reset register
0x0040	RCC_APB2RSTR	APB2 peripheral reset register
0x0048	RCC_AHB1ENR	AHB1 peripheral enable register
0x004c	RCC_AHB2ENR	AHB2 peripheral enable register
0x0050	RCC_AHB2ENR1	AHB2 peripheral enable register 1
0x0058	RCC_APB1ENR	APB1 peripheral enable register

Offset	Register Name	Register Description
0x0060	RCC_APB2ENR	APB2 peripheral enable register
0x0070	RCC_PLLCFGR3	PLL configuration register 3
0x0074	RCC_PLLCFGR4	PLL configuration register 4
0x0088	RCC_CCIPR	Peripherals independent clock configuration register
0x0094	RCC_CSR	Control/status register
0x0098	RCC_CFG2R	Clock configuration register 2
0x009c	RCC_CFG3R	Clock configuration register 3
0x00a0	RCC_BDCR	RTC domain control register
0x00a4	RCC_BDCR2	RTC domain control register 2
0x00a8	RCC_BDCR3	RTC domain control register 3
0x00ac	RCC_BDCR4	RTC domain control register 4
0x00b8	RCC_BDCR5	RTC domain control register 5
0x00bc	RCC_BDCR6	RTC domain control register 6
0x00c0	RCC_BDCR7	RTC domain control register 7

5.5.2 Register Field Details

5.5.2.1 RCC_CR

0x0000			Clock control register											RCC_CR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved						PLLRDY	PLLON	Reserved				CSSON	EHSBYP	EHSRDY	EHSON
Type	RO						RO	RW	RO				RW	RW	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EHSECLKON	Reserved				IHSRDY	Reserved	IHSON	Reserved							
Type	RW	RO				RO	RO	RW	RO							
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 5-2 Clock Control Register Description

Field	Name	Description
31:26	Reserved	Reserved
25	PLLRDY	Set by hardware to indicate that the main PLL is locked. 0: PLL unlocked 1: PLL locked
24	PLLON	Set and cleared by software to enable the main PLL. Cleared by hardware when entering stop, standby or shutdown mode. This bit cannot be reset if the PLL clock is used as the system clock. 0: PLL OFF 1: PLL ON

Field	Name	Description
23:20	Reserved	Reserved
19	CSSON	Set by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the EHS oscillator is ready, and disabled by hardware if an EHS clock failure is detected. This bit is set only and is cleared by reset. 0: Clock security system OFF (clock detector OFF) 1: Clock security system ON (Clock detector ON if the EHS oscillator is stable, OFF if not).
18	EHSBYP	Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the EHSON bit set, to be used by the device. The EHSBYP bit can be written only if the EHS oscillator is disabled. 0: EHS crystal oscillator not bypassed 1: EHS crystal oscillator bypassed with external clock
17	EHSRDY	Set by hardware to indicate that the EHS oscillator is stable. 0: EHS oscillator not ready 1: EHS oscillator ready
16	EHSON	Set and cleared by software. Cleared by hardware to stop the EHS oscillator when entering standby or shutdown mode. This bit cannot be reset if the EHS oscillator is used directly or indirectly as the system clock. 0: EHS oscillator OFF 1: EHS oscillator ON
15	EHSECLKON	Set and cleared by software. Cleared by hardware to stop the EHS oscillator when entering standby or shutdown mode. 0: EHS clk output to external OFF 1: EHS clk output to external ON
14:11	Reserved	Reserved

Field	Name	Description
10	IHSRDY	Set by hardware to indicate that IHS oscillator is stable. This bit is set only when IHS is enabled by software by setting IHSON. 0: IHS oscillator not ready 1: IHS oscillator ready
9	Reserved	Reserved
8	IHSON	Set and cleared by software. Cleared by hardware to stop the IHS oscillator when entering standby1, standby2 or shutdown mode. Set by hardware to force the IHS oscillator ON in case of failure of the EHS crystal oscillator. This bit is set by hardware if the IHS is used directly or indirectly as system clock. 0: IHS oscillator OFF 1: IHS oscillator ON
7:0	Reserved	Reserved

5.5.2.2 RCC_CR1

0x0004			Clock control register 1											RCC_CR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															EHSTESTEN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EHSCTRL								EHSFREQSEL							
Type	RW								RW							
Reset	0	0	0	0	1	0	1	1	0	0	0	0	0	1	0	0

Table 5-3 Clock Control Register 1 Description

Field	Name	Description
31:17	Reserved	Reserved
16	EHSTESTEN	1: Enable DC test mux and clock test buffer
15:8	EHSCTRL	The 2 MSBs EHSCTRL[7:6] are EHS_READY_GATING (EHSCTRL[7]) and EHS_READY_SET (EHSCTRL[6]) respectively. EHSCTRL[7:6]=2'b00: Normal mode, EHS_READY is determined by circuit state. EHSCTRL[7:6]=2'b01: EHS_READY = 1 (if PD = 1, EHS_READY is still equal to 0). EHSCTRL[7:6]=2'b10: EHS_READY=0 EHSCTRL[7:6]=2'b11: Forbidden mode. In this mode, EHS_READY = 0. The left 6 LSBs EHSCTRL[5:0] are current control bits of ehs core. Current control bits of ehs core. Steady state: Slower than 25M EHSCTRL = 6'd11; faster than 25M EHSCTRL = 6'd27.

Field	Name	Description
		After power up, for fast startup: Slower than 25M EHSCTRL = 6'd40; faster than 25M EHSCTRL = 6'd63. If EHS_READY gets high, EHSCTRL can change back to steady state value (decided by user). The detail dig_control timing is shown in 'EHS_dig_control timing' sheet.
7:0	EHSFREQSEL	The 4 MSBs are reserved. EHSFREQSEL[3] is ehs_bias_bk_mode. EHSFREQSEL[3]=1: the bias current is from local backup bias. EHSFREQSEL[3]=0: the bias current is from PMU. EHSFREQSEL[2:0] are EHS_READY counter's pre_divider. EHSFREQSEL[2:0]=3'd0: pre_divider_ratio=1 EHSFREQSEL[2:0]=3'd1: pre_divider_ratio=2 EHSFREQSEL[2:0]=3'd2: pre_divider_ratio=4 ..... EHSFREQSEL[2:0]=3'd7: pre_divider_ratio=128 Default setting for EHSFREQSELL[2:0]: frequency= 16~32MHz: EHSFREQSEL[2:0]=3'd5 frequency=8~16MHz: EHSFREQSEL[2:0]=3'd4 frequency=4~8MHz: EHSFREQSEL[2:0]=3'd3 This default value may be changed according to test results.

5.5.2.3 RCC_CFGR

0x0008			Clock configuration register											RCC_CFGR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				MCOEN	MCOSEL			MCOPRE							
Type	RO				RW	RW			RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		PPRE2			PPRE1			HPRE				SWS	SW		
Type	RO		RW			RW			RW				RO	RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-4 Clock Configuration Register Description

Field	Name	Description
31:28	Reserved	Reserved
27	MCOEN	MCO CLK enable 0: Disable 1: Enable
26:24	MCOSEL	Microcontroller clock output Set and cleared by software. 000: SYSCLK system clock selected 001: IHS clock selected 010: EHS clock selected 011: Main PLL clock selected 100: ILS clock selected 101: ELS clock selected 110: ADC PLL clock selected

Field	Name	Description
		111: TPC_CK_HP clock selected **NOTE:** This clock output may have some truncated cycles at startup or during MCO clock source switching.
23:16	MCOPRE	Microcontroller clock output prescaler These bits are set and cleared by software. It is highly recommended to change this prescaler before MCO output is enabled. 00000000: MCO is divided by 1 00000001: MCO is divided by 2 00000010: MCO is divided by 3 00000011: MCO is divided by 4 00000100: MCO is divided by 5 ... 11111111: MCO is divided by 256
15:14	Reserved	Reserved
13:11	PPRE2	Set and cleared by software to control the division factor of the APB2 clock (PCLK2). 0xx: HCLK not divided 100: HCLK divided by 2 101: HCLK divided by 4 110: HCLK divided by 8 111: HCLK divided by 16
10:8	PPRE1	Set and cleared by software to control the division factor of the APB1 clock (PCLK1). 0xx: HCLK not divided 100: HCLK divided by 2 101: HCLK divided by 4 110: HCLK divided by 8 111: HCLK divided by 16

Field	Name	Description
7:4	HPRE	AHB prescaler 0xxx: SYSCLK not divided 1000: SYSCLK divided by 2 1001: SYSCLK divided by 4 1010: SYSCLK divided by 8 1011: SYSCLK divided by 16 1100: SYSCLK divided by 64 1101: SYSCLK divided by 128 1110: SYSCLK divided by 256 1111: SYSCLK divided by 512
3:2	SWS	Set and cleared by hardware to indicate which clock source is used as system clock. 00: IHS oscillator used as system clock 01: EHS used as system clock 10: PLL used as system clock 11: Reserved, must be kept at reset value
1:0	SW	Set and cleared by software to select system clock source (SYSCLK). Configured by hardware to force IHS oscillator selection when exiting standby and shutdown modes or in case of failure of the EHS oscillator. 00: IHS selected as system clock 01: EHS selected as system clock 10: PLL selected as system clock 11: Reserved, must be kept at reset value

5.5.2.4 RCC_PLLCFGR1

0x000c			PLL configuration register											RCC_PLLCFGR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		PLLFREFLEAD	PLLDIV48M		PLLDIV48MPD	Reserved									
Type	RO		RO	RW		RW	RO									
Reset	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	PLLODIV		PLLFBDIV								PLLINDIV		PLLMANUALRSTN	PLLSRC	
Type	RO	RW		RW								RW		RW	RW	
Reset	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0

Table 5-5 PLL Configuration Register Description

Field	Name	Description
31:30	Reserved	Reserved
29	PLLFREFLEAD	PLL Fref Lead If the input reference signal frequency (F_{ref}) is greater than the feedback signal frequency (F_{fb}), this signal will become high.
28:27	PLLDIV48M	00: DIV2 01: DIV4 10: DIV8 11: Reserved
26	PLLDIV48MPD	PLLDIV48M output clock pd control:

Field	Name	Description
		0: Turn on DIV4 if PLLON = 1, it will not valid if PLLON = 0 1: Turn off DIV4
25:15	Reserved	Reserved
14:13	PLLODIV	00: DIV2 01: DIV4 10: DIV8 11: Reserved
12:5	PLLFBDIV	Divider value valid range: 7~180
4:3	PLLINDIV	Input divider 00: DIV1 01: DIV2 10: DIV4 11: DIV8
2	PLLMANUALRSTN	Manually restart PLL
1:0	PLLSRC	Main PLL entry clock source: Set and cleared by software to select PLL clock source. These bits can be written only when PLL is disabled. To save power, when no PLL is used, the value of PLLSRC should be 00. 00: No clock sent to PLL 01: No clock sent to PLL 10: IHS clock selected as PLL clock entry 11: EHS clock selected as PLL clock entry

5.5.2.5 RCC_PLLCFGR2

0x0010			PLL configuration register 2											RCC_PLLCFGR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved	PLLRDIVFRAC		PLLRDIVINT				PLLRDIVCTRL								PLLRDIVPD
Type	RO	RW		RW				RW								RW
Reset	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PLLCTRL1								PLLCTRL0							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1

Table 5-6 PLL Configuration Register 2 Description

Field	Name	Description
31	Reserved	Reserved
30:29	PLLRDIVFRAC	RDIV frac div ratio part: 0:0.25:0.75. FVCO=~360M: 0x1; (CLK_OUT ~180M) FVCO=~288M: 0x2; (CLK_OUT ~72M) FVCO=~240M: 0x0; (CLK_OUT ~120M) FVCO=~192M: 0x0; (CLK_OUT ~92M 48M 24M)
28:25	PLLRDIVINT	RDIV int div ratio part: 4~15. FVCO=~360M: 0xd; (CLK_OUT ~180M) FVCO=~288M: 0xa; (CLK_OUT ~72M) FVCO=~240M: 0x9; (CLK_OUT ~120M) FVCO=~192M: 0x7; (CLK_OUT ~92M 48M 24M)

Field	Name	Description
24:17	PLLRDIVCTRL	Reserved
16	PLLRDIVPD	PLLRDIV output clock pd control: 0: Turn on RDIV if PLLON = 1; it will not be valid if PLLON = 0 1: Turn off RDIV
15:8	PLLCTRL1	Private register
7:0	PLLCTRL0	Private register

5.5.2.6 RCC_CR2

0x0014			Clock control register 2											RCC_CR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				EHSCSSTARGETCLKCNTEND											
Type	RO				RW											
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				EHSCSSLOCALCLKCNTEND											
Type	RO				RW											
Reset	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0

Table 5-7 Clock Control Register 2 Description

Field	Name	Description
31:28	Reserved	Reserved
27:16	EHSCSSTARGETCLKCNTEND	ehs_css clk_ehs counter
15:12	Reserved	Reserved
11:0	EHSCSSLOCALCLKCNTEND	ehs_css clk_ihs counter This counter will restart from 0 when clk_ehs counter reach ehs_css_target_clk_cnt_end. If clk_ihs counter reach ehs_css_local_clk_cnt_end, css fail detected.

5.5.2.7 RCC_CIER

0x0018			Clock interrupt enable register											RCC_CIER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved									ELSCSSIE	Reserved	PLLRDYIE	EHSRDYIE	IHSRDYIE	ELSRDYIE	ILSRDYIE
Type	RO									RW	RO	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-8 Clock Interrupt Enable Register Description

Field	Name	Description
31:7	Reserved	Reserved
6	ELSCSSIE	ELS clock security system interrupt enable Set and cleared by software to enable/disable interrupt caused by the clock security system on ELS. 0: Clock security interrupt caused by ELS clock failure disabled 1: Clock security interrupt caused by ELS clock failure enabled
5:5	Reserved	Reserved
4	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
3	EHSRDYIE	EHS ready interrupt enable

Field	Name	Description
		Set and cleared by software to enable/disable interrupt caused by the EHS oscillator stabilization. 0: EHS ready interrupt disabled 1: EHS ready interrupt enabled
2	IHSRDYIE	IHS ready interrupt enable Set and cleared by software to enable/disable interrupt caused by the IHS oscillator stabilization. 0: IHS ready interrupt disabled 1: IHS ready interrupt enabled
1	ELSRDYIE	ELS ready interrupt enable Set and cleared by software to enable/disable interrupt caused by the ELS oscillator stabilization. 0: ELS ready interrupt disabled 1: ELS ready interrupt enabled
0	ILSRDYIE	ILS ready interrupt enable Set and cleared by software to enable/disable interrupt caused by the ILS oscillator stabilization. 0: ILS ready interrupt disabled 1: ILS ready interrupt enabled

5.5.2.8 RCC_CIFR

0x001c			Clock interrupt flag register											RCC_CIFR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved									ELSCSSF	CSSF	PLLRDYF	EHSRDYF	IHSRDYF	ELSRDYF	ILSRDYF
Type	RO									RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-9 Clock Interrupt Flag Description

Field	Name	Description
31:7	Reserved	Reserved
6	ELSCSSF	ELS Clock security system interrupt flag Set by hardware when a failure is detected in the ELS oscillator. Cleared by software setting the ELSCSSC bit. 0: No clock security interrupt caused by ELS clock failure 1: Clock security interrupt caused by ELS clock failure
5	CSSF	Clock security system interrupt flag: Set by hardware when a failure is detected in the EHS oscillator. Cleared by software setting the CSSC bit. 0: No clock security interrupt caused by EHS clock failure 1: Clock security interrupt caused by EHS clock failure

Field	Name	Description
4	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	EHSRDYF	EHS ready interrupt flag Set by hardware when the EHS clock becomes stable and EHSRDYDIE is set. Cleared by software setting the EHSRDYC bit. 0: No clock ready interrupt caused by the EHS oscillator 1: Clock ready interrupt caused by the EHS oscillator
2	IHSRDYF	IHS ready interrupt flag Set by hardware when the IHS clock becomes stable, and HSIRDYDIE is set in response to setting the IHSON in the RCC_CR register. If IHSON is not set, but the IHS oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated. Cleared by software setting the HSIRDYC bit. 0: No clock ready interrupt caused by the IHS oscillator 1: Clock ready interrupt caused by the IHS oscillator
1	ELSRDYF	ELS ready interrupt flag Set by hardware when the ELS clock becomes stable and ELSRDYDIE is set. Cleared by software setting the ELSRDYC bit. 0: No clock ready interrupt caused by the ELS oscillator 1: Clock ready interrupt caused by the ELS oscillator
0	ILSRDYF	ILS ready interrupt flag Set by hardware when the ILS clock becomes stable and ILSRDYDIE is set. Cleared by software setting the ILSRDYC bit. 0: No clock ready interrupt caused by the ILS oscillator

Field	Name	Description
		1: Clock ready interrupt caused by the ILS oscillator

5.5.2.9 RCC_CICR

0x0020			Clock interrupt clear register											RCC_CICR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved									ELSCSSC	CSSC	PLLRDYC	EHSRDYC	IHSRDYC	ELSRDYC	ILSRDYC
Type	RO									WC	WC	WC	WC	WC	WC	WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-10 Clock Interrupt Clear Register Description

Field	Name	Description
31:7	Reserved	Reserved
6	ELSCSSC	ELS Clock security system interrupt clear This bit is set by software to clear the ELSCSSF flag. 0: No effect 1: Clear ELSCSSF flag
5	CSSC	Clock security system interrupt clear This bit is set by software to clear the CSSF flag. 0: No effect 1: Clear CSSF flag
4	PLLRDYC	PLL ready interrupt clear This bit is set by software to clear the PLLRDYF flag. 0: No effect

Field	Name	Description
		1: Clear PLLRDYF flag
3	EHSRDYC	EHS ready interrupt clear This bit is set by software to clear the EHSRDYF flag. 0: No effect 1: Clear EHSRDYF flag
2	IHSRDYC	IHS ready interrupt clear This bit is set software to clear the HSIRDYF flag. 0: No effect 1: Clear HSIRDYF flag
1	ELSRDYC	ELS ready interrupt clear This bit is set by software to clear the ELSRDYF flag. 0: No effect 1: ELSRDYF cleared
0	ILSRDYC	ILS ready interrupt clear This bit is set by software to clear the ILSRDYF flag. 0: No effect 1: ILSRDYF cleared

5.5.2.10 RCC_AHB1RSTR

0x0028			AHB1 peripheral reset register											RCC_AHB1RSTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
Type	RO										RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-11 AHB1 Peripheral Reset Register Description

Field	Name	Description
31:6	Reserved	Reserved
5	GPIOFRST	IO port F reset Set and cleared by software. 0: No effect 1: Reset IO port F
4	GPIOERST	IO port E reset Set and cleared by software. 0: No effect 1: Reset IO port E
3	GPIODRST	IO port D reset Set and cleared by software. 0: No effect

Field	Name	Description
		1: Reset IO port D
2	GPIOCRST	IO port C reset Set and cleared by software. 0: No effect 1: Reset IO port C
1	GPIOBRST	IO port B reset Set and cleared by software. 0: No effect 1: Reset IO port B
0	GPIOARST	IO port A reset Set and cleared by software. 0: No effect 1: Reset IO port A

5.5.2.11 RCC_AHB2RSTR

0x002c			AHB2 peripheral reset register											RCC_AHB2RSTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					ADCRST	Reserved					HASHRST	AESRST	CRCRST	DMA1RST	DMA0RST
Type	RO					RW	RO					RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-12 AHB2 Peripheral Reset Register Description

Field	Name	Description
31:11	Reserved	Reserved
10	ADCRST	ADC reset Set and cleared by software. 0: No effect 1: Reset ADC
9:5	Reserved	Reserved
4	HASHRST	HASH reset Set and cleared by software. 0: No effect 1: Reset HASH
3	AESRST	AES reset Set and cleared by software.

Field	Name	Description
		0: No effect 1: Reset AES
2	CRCRST	CRC reset Set and cleared by software. 0: No effect 1: Reset CRC
1	DMA1RST	DMA1 reset Set and cleared by software. 0: No effect 1: Reset DMA1
0	DMA0RST	DMA0 reset Set and cleared by software. 0: No effect 1: Reset DMA0

5.5.2.12 RCC_AHB2RSTR1

0x0030			AHB2 peripheral reset register 1											RCC_AHB2RSTR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														FLASH RST	Reserved
Type	RO														RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-13 AHB2 Peripheral Reset Register 1 Description

Field	Name	Description
31:2	Reserved	Reserved
1	FLASHRST	Flash reset Set and cleared by software. 0: No effect 1: Reset Flash interface
0	Reserved	Reserved

5.5.2.13 RCC_APB1RSTR

0x0038			APB1 peripheral reset register											RCC_APB1RSTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															PMURST
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	USART6RST	SPI1RST	SPI0RST	DACRST	WWDGRST	USART5RST	USART4RST	I2C1RST	I2C0RST	RTCAPBRST	GPTMR5RST	GPTMR4RST	GPTMR3RST	GPTMR2RST	GPTMR1RST	GPTMR0RST
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-14 APB1 Peripheral Reset Register Description

Field	Name	Description
31:17	Reserved	Reserved
16	PMURST	PMU reset Set and cleared by software. 0: No effect 1: Reset PMU interface
15	USART6RST	USART6 reset Set and cleared by software. 0: No effect 1: Reset USART6
14	SPI1RST	SPI1 reset Set and cleared by software.

Field	Name	Description
		0: No effect 1: Reset SPI1
13	SPI0RST	SPI0 reset Set and cleared by software. 0: No effect 1: Reset SPI0
12	DACRST	DAC reset Set and cleared by software. 0: No effect 1: Reset DAC
11	WWDGRST	WWDG reset Set and cleared by software. 0: No effect 1: Reset WWDG
10	USART5RST	USART5 reset Set and cleared by software. 0: No effect 1: Reset USART5
9	USART4RST	USART4 reset Set and cleared by software. 0: No effect 1: Reset USART4
8	I2C1RST	I2C1 reset Set and cleared by software. 0: No effect 1: Reset I2C1

Field	Name	Description
7	I2C0RST	I2C0 reset Set and cleared by software. 0: No effect 1: Reset I2C0
6	RTCAPBRST	RTC interface reset Set and cleared by software. 0: No effect 1: Reset RTC interface
5	GPTMR5RST	GPTMR5 reset Set and cleared by software. 0: No effect 1: Reset GPTMR5
4	GPTMR4RST	GPTMR4 reset Set and cleared by software. 0: No effect 1: Reset GPTMR4
3	GPTMR3RST	GPTMR3 reset Set and cleared by software. 0: No effect 1: Reset GPTMR3
2	GPTMR2RST	GPTMR2 reset Set and cleared by software. 0: No effect 1: Reset GPTMR2
1	GPTMR1RST	GPTMR1 reset Set and cleared by software.

Field	Name	Description
		0: No effect 1: Reset GPTMR1
0	GPTMR0RST	GPTMR0 reset Set and cleared by software. 0: No effect 1: Reset GPTMR0

5.5.2.14 RCC_APB2RSTR

0x0040			APB2 peripheral reset register											RCC_APB2RSTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															TRNGRST
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OA1RST	OA0RST	CMP1RST	CMP0RST	VREFBUFRST	SPI2RST	CANRST	ADVTMR2RST	ADVTMR1RST	ADVTMR0RST	SYSCFGRST	USART3RST	USART2RST	USART1RST	USART0RST	TPCRST
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-15 APB2 Peripheral Reset Register Description

Field	Name	Description
31:17	Reserved	Reserved
16	TRNGRST	trng reset Set and cleared by software. 0: No effect 1: Reset trng
15	OA1RST	OA1 reset Set and cleared by software. 0: No effect 1: Reset OA1
14	OA0RST	OA0 reset Set and cleared by software. 0: No effect 1: Reset OA0
13	CMP1RST	CMP1 reset Set and cleared by software.

Field	Name	Description
		0: No effect 1: Reset CMP1
12	CMP0RST	CMP0 reset Set and cleared by software. 0: No effect 1: Reset CMP0
11	VREFBUFRST	VREFBUF reset Set and cleared by software. 0: No effect 1: Reset VREFBUF
10	SPI2RST	SPI2 reset Set and cleared by software. 0: No effect 1: Reset SPI2
9	CANRST	CAN reset Set and cleared by software. 0: No effect 1: Reset CAN
8	ADVTMR2RST	ADVTMR2 reset Set and cleared by software. 0: No effect 1: Reset ADVTMR2
7	ADVTMR1RST	ADVTMR1 reset Set and cleared by software. 0: No effect 1: Reset ADVTMR1
6	ADVTMR0RST	ADVTMR0 reset Set and cleared by software. 0: No effect 1: Reset ADVTMR0
5	SYSCFGRST	SYSCFG reset Set and cleared by software. 0: No effect

Field	Name	Description
		1: Reset SYSCFG
4	USART3RST	USART3 reset Set and cleared by software. 0: No effect 1: Reset USART3
3	USART2RST	USART2 reset Set and cleared by software. 0: No effect 1: Reset USART2
2	USART1RST	USART1 reset Set and cleared by software. 0: No effect 1: Reset USART1
1	USART0RST	USART0 reset Set and cleared by software. 0: No effect 1: Reset USART0
0	TPCRST	TPC reset Set and cleared by software. 0: No effect 1: Reset TPC

5.5.2.15 RCC_AHB1ENR

0x0048			AHB1 peripheral enable register											RCC_AHB1ENR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
Type	RO										RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-16 AHB1 Peripheral Enable Register Description

Field	Name	Description
31:6	Reserved	Reserved
5	GPIOFEN	IO port F clock enable Set and cleared by software. 0: IO port F clock disabled 1: IO port F clock enabled
4	GPIOEEN	IO port E clock enable Set and cleared by software. 0: IO port E clock disabled 1: IO port E clock enabled
3	GPIODEN	IO port D clock enable Set and cleared by software.

Field	Name	Description
		0: IO port D clock disabled 1: IO port D clock enabled
2	GPIOCEN	IO port C clock enable Set and cleared by software. 0: IO port C clock disabled 1: IO port C clock enabled
1	GPIOBEN	IO port B clock enable Set and cleared by software. 0: IO port B clock disabled 1: IO port B clock enabled
0	GPIOAEN	IO port A clock enable Set and cleared by software. 0: IO port A clock disabled 1: IO port A clock enabled

5.5.2.16 RCC_AHB2ENR

0x004c			AHB2 peripheral enable register											RCC_AHB2ENR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										ADCEN	HASHEN	AESEN	CRCEN	DMA1EN	DMA0EN
Type	RO										RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-17 AHB2 pPeripheral Enable Register Description

Field	Name	Description
31:6	Reserved	Reserved
5	ADCEN	ADC clock enable Set and cleared by software. 0: ADC clock disabled 1: ADC clock enabled
4	HASHEN	HASH clock enable Set and cleared by software. 0: HASH clock disabled 1: HASH clock enabled
3	AESEN	AES clock enable Set and cleared by software. 0: AES clock disabled

Field	Name	Description
		1: AES clock enabled
2	CRCEN	CRC clock enable Set and cleared by software. 0: CRC clock disabled 1: CRC clock enabled
1	DMA1EN	DMA1 clock enable Set and cleared by software. 0: DMA1 clock disabled 1: DMA1 clock enabled
0	DMA0EN	DMA0 clock enable Set and cleared by software. 0: DMA0 clock disabled 1: DMA0 clock enabled

5.5.2.17 RCC_AHB2ENR1

0x0050			AHB2 peripheral enable register 1											RCC_AHB2ENR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														FLASH EN	Reserved
Type	RO														RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 5-18 AHB2 Peripheral Enable Register 1 Description

Field	Name	Description
31:2	Reserved	Reserved
1	FLASHEN	Flash clock enable Set and cleared by software. 0: Flash clock disabled 1: Flash clock enabled
0	Reserved	Reserved

5.5.2.18 RCC_APB1ENR

0x0058			APB1 peripheral enable register											RCC_APB1ENR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															PMUEN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	USART 6EN	SPI1EN	SPI0EN	DACEN	WWDG EN	USART 5EN	USART 4EN	I2C1EN	I2C0EN	RTCAP BEN	GPTMR 5EN	GPTMR 4EN	GPTMR 3EN	GPTMR 2EN	GPTMR 1EN	GPTMR 0EN
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-19 APB1 Peripheral Enable Register Description

Field	Name	Description
31:17	Reserved	Reserved
16	PMUEN	PMU APB clock enable Set and cleared by software. 0: PMU APB clock disabled 1: PMU APB clock enabled
15	USART6EN	USART6 clock enable Set and cleared by software. 0: USART6 clock disabled 1: USART6 clock enabled
14	SPI1EN	SPI1 clock enable Set and cleared by software.

Field	Name	Description
		0: SPI1 clock disabled 1: SPI1 clock enabled
13	SPI0EN	SPI0 clock enable Set and cleared by software. 0: SPI0 clock disabled 1: SPI0 clock enabled
12	DACEN	DAC clock enable Set and cleared by software. 0: DAC clock disabled 1: DAC clock enabled
11	WWDGEN	WWDG clock enable Set and cleared by software. 0: WWDG clock disabled 1: WWDG clock enabled
10	USART5EN	USART5 clock enable Set and cleared by software. 0: USART5 clock disabled 1: USART5 clock enabled
9	USART4EN	USART4 clock enable Set and cleared by software. 0: USART4 clock disabled 1: USART4 clock enabled
8	I2C1EN	I2C1 clock enable Set and cleared by software. 0: I2C1 clock disabled 1: I2C1 clock enabled

Field	Name	Description
7	I2C0EN	I2C0 clock enable Set and cleared by software. 0: I2C0 clock disabled 1: I2C0 clock enabled
6	RTCAPBEN	RTC APB clock enable Set and cleared by software. 0: RTC APB clock disabled 1: RTC APB clock enabled
5	GPTMR5EN	GPTMR5 clock enable Set and cleared by software. 0: GPTMR5 clock disabled 1: GPTMR5 clock enabled
4	GPTMR4EN	GPTMR4 clock enable Set and cleared by software. 0: GPTMR4 clock disabled 1: GPTMR4 clock enabled
3	GPTMR3EN	GPTMR3 clock enable Set and cleared by software. 0: GPTMR3 clock disabled 1: GPTMR3 clock enabled
2	GPTMR2EN	GPTMR2 clock enable Set and cleared by software. 0: GPTMR2 clock disabled 1: GPTMR2 clock enabled
1	GPTMR1EN	GPTMR1 clock enable Set and cleared by software.

Field	Name	Description
		0: GPTMR1 clock disabled 1: GPTMR1 clock enabled
0	GPTMR0EN	GPTMR0 clock enable Set and cleared by software. 0: GPTMR0 clock disabled 1: GPTMR0 clock enabled

5.5.2.19 RCC_APB2ENR

0x0060			APB2 peripheral enable register											RCC_APB2ENR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															TRNGEN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OA1EN	OA0EN	CMP1EN	CMP0EN	VREFBUFEN	SPI2EN	CANEN	ADVTMR2EN	ADVTMR1EN	ADVTMR0EN	SYSCFGEN	USART3EN	USART2EN	USART1EN	USART0EN	TPCEN
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-20 APB2 Peripheral Enable Register Description

Field	Name	Description
31:17	Reserved	Reserved
16	TRNGEN	TRNG clock enable Set and cleared by software. 0: TRNG clock disabled 1: TRNG clock enabled
15	OA1EN	OA1 clock enable Set and cleared by software. 0: OA1 clock disabled 1: OA1 clock enabled
14	OA0EN	OA0 clock enable Set and cleared by software.

Field	Name	Description
		0: OA0 clock disabled 1: OA0 clock enabled
13	CMP1EN	CMP1 clock enable Set and cleared by software. 0: CMP1 clock disabled 1: CMP1 clock enabled
12	CMP0EN	CMP0 clock enable Set and cleared by software. 0: CMP0 clock disabled 1: CMP0 clock enabled
11	VREFBUFEN	VREFBUF clock enable Set and cleared by software. 0: VREFBUF clock disabled 1: VREFBUF clock enabled
10	SPI2EN	SPI2 clock enable Set and cleared by software. 0: SPI2 clock disabled 1: SPI2 clock enabled
9	CANEN	CAN clock enable Set and cleared by software. 0: CAN clock disabled 1: CAN clock enabled
8	ADVTMR2EN	ADVTMR2 clock enable Set and cleared by software. 0: ADVTMR2 clock disabled 1: ADVTMR2 clock enabled

Field	Name	Description
7	ADVTMR1EN	ADVTMR1 clock enable Set and cleared by software. 0: ADVTMR1 clock disabled 1: ADVTMR1 clock enabled
6	ADVTMR0EN	ADVTMR0 clock enable Set and cleared by software. 0: ADVTMR0 clock disabled 1: ADVTMR0 clock enabled
5	SYSCFGEN	SYSCFG clock enable Set and cleared by software. 0: SYSCFG clock disabled 1: SYSCFG clock enabled
4	USART3EN	USART3 clock enable Set and cleared by software. 0: USART3 clock disabled 1: USART3 clock enabled
3	USART2EN	USART2 clock enable Set and cleared by software. 0: USART2 clock disabled 1: USART2 clock enabled
2	USART1EN	USART1 clock enable Set and cleared by software. 0: USART1 clock disabled 1: USART1 clock enabled
1	USART0EN	USART0 clock enable Set and cleared by software.

Field	Name	Description
		0: USART0 clock disabled 1: USART0 clock enabled
0	TPCEN	TPC clock enable Set and cleared by software. 0: TPC clock disabled 1: TPC clock enabled

5.5.2.20 RCC_PLLCFGR3

0x0070			PLL configuration register 3												RCC_PLLCFGR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Name	Reserved	PLLPFDCPCTRL					PLLLPF3RDBYPASS	PLLLPFR3					PLLLPFR2				
Type	RO	RW					RW	RW					RW				
Reset	0	1	0	1	0	0	0	0	1	0	0	1	0	1	0	0	
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	PLLLPFR2	PLLLPFC3					PLLLPFC2					PLLLPFC1					
Type	RW	RW					RW					RW					
Reset	1	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	

Table 5-21 PLL Configuration Register 3 Description

Field	Name	Description
31	Reserved	Reserved
30:26	PLLPFDCPCTRL	[2:0] cp current control words default: 3'd4; 6uA~13uA 1uA/step. [4:3] pfd dead zone time control words default: 2'd1;250ps/500ps/750ps/1ns
25	PLLLPF3RDBYPASS	LPF R3 and C3 bypass control. 0: Disable (3rd order LPF mode) 1: Enable (2nd order LPF mode)
24:20	PLLLPFR3	LPF R3 control words. setting1: PFD freq 4~8M setting2: PFD freq 8~32M setting1: 0x09 setting2: 0x09 The default value is setting2.

Field	Name	Description
19:15	PLLLPFR2	LPF R2 control words. setting1: PFD freq 4~8M setting2: PFD freq 8~32M setting1: 0x09 setting2: 0x09 The default value is setting2.
14:10	PLLLPFC3	LPF C3 control words. setting1: PFD freq 4~8M setting2: PFD freq 8~32M setting1: 0x14 setting2: 0x02 The default value is setting2.
9:5	PLLLPFC2	LPF C2 control words. setting1: PFD freq 4~8M setting2: PFD freq 8~32M setting1: 0x0A setting2: 0x08 The default value is setting2.
4:0	PLLLPFC1	LPF C1 control words. setting1: PFD freq 4~8M setting2: PFD freq 8~32M setting1: 0x14 setting2: 0x04 The default value is setting2.

5.5.2.21 RCC_PLLCFGR4

0x0074			PLL configuration register 4											RCC_PLLCFGR4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															PLLVCOCTRL
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PLLVCOCTRL				PLLCLKTSTOL	PLLCLKTSTCTRL		PLLLOCKEDCTRL				PLLTESTMUX				PLLTESTEN
Type	RW				RW	RW		RW				RW				RW
Reset	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0

Table 5-22 PLL Configuration Register 4 Description

Field	Name	Description
31:17	Reserved	Reserved
16:12	PLLVCOCTRL	VCO band control words: 350-360M 0x13 330M-350M 0x11 280M-330M 0xC 180M-280M 0x4
11	PLLCLKTSTOL	0: PLL in close loop 1: PLL in open loop for freq cover range test
10:9	PLLCLKTSTCTRL	00: CLK_OUT 01: CLK_RDIV_OUT

Field	Name	Description
		10: CLK_DIV48M_OUT 11: CLK_FBDIV_OUT
8:5	PLLLOCKEDCTRL	PLLLOCKEDCTRL[3:2] is mode control: 00: Set, output = 1 01: Single run (if failed, it will continue running till success) 10: Continuous 11: Power down, output = 0 PLLLOCKEDCTRL[1:0] is counter setting: 256/512/1024/2048 respectively.
4:1	PLLTESTMUX	DC test control words 0000: vbp of cp 0001: vbn of cp 0010: vtune 0011: vbn of vco_v2i 0100: vbp of vco_v2i 0101: vbp of vco_bias 0110: vrefp of vco_bias 0111: vrefn of vco_bias 1xxx: lockdet
0	PLLTESTEN	0: Disable test mode 1: Enable test mode

5.5.2.22 RCC_CSR

0x0094			Control/status register											RCC_CSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	Reserved	RMVF	Reserved						
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RO						
Reset	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														LSCOSEL	LSCOEN
Type	RO														RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-23 Control/Status Register Description

Field	Name	Description
31	LPWRRSTF	Low-power reset flag Set by hardware when a reset occurs due to illegal stop, standby or shutdown mode entry. Cleared by writing to the RMVF bit. 0: No illegal mode reset occurred 1: Illegal mode reset occurred
30	WWDGRSTF	Window Watchdog (WWDG) reset flag Set by hardware when a WWDG reset occurs. Cleared by writing to the RMVF bit. 0: No WWDG reset occurred 1: WWDG reset occurred
29	IWDGRSTF	Independent Watchdog (IWDG) reset flag

Field	Name	Description
		Set by hardware when an IWDG reset domain occurs. Cleared by writing to the RMVF bit. 0: No IWDG reset occurred 1: IWDG reset occurred
28	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	BORRSTF	BOR flag Set by hardware when a BOR occurs. Cleared by writing to the RMVF bit. 0: No BOR occurred 1: BOR occurred
26	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	OBLRSTF	Option byte loader reset flag Set by hardware when a reset from the Option Byte loading occurs. Cleared by writing to the RMVF bit. 0: No reset from Option Byte loading occurred 1: Reset from Option Byte loading occurred
24:24	Reserved	Reserved
23	RMVF	Remove reset flag

Field	Name	Description
		Set by software to clear the reset flags. 0: No effect 1: Clear the reset flags
22:2	Reserved	Reserved
1	LSCOSEL	Low-speed clock output selection Set and cleared by software. 0: ILS clock selected 1: ELS clock selected
0	LSCOEN	Low-speed clock output enable Set and cleared by software. 0: Low-speed clock output (LSCO) disable 1: Low-speed clock output (LSCO) enable

5.5.2.23 RCC_CCIPR

0x0088			Peripherals independent clock configuration register											RCC_CCIPR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										PMUSEL		TPCSEL		ADCSEL	
Type	RO										RW		RW		RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CANSEL		USART6SEL		USART3SEL		USART2SEL		USART1SEL		USART0SEL		USART5SEL		USART4SEL	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-24 Peripherals Independent Clock Configuration Register Description

Field	Name	Description
31:22	Reserved	Reserved
21:20	PMUSEL	2'b00: IHS 2'b10: ILS 2'b11: ELS
19:18	TPCSEL	TPC clock source selection These bits are set and cleared by software to select the TPC clock source. 00: PLL clock selected as TPC clock 01: IHS clock selected as TPC clock 10: EHS clock selected as TPC clock
17:16	ADCSEL	ADC clock source selection These bits are set and cleared by software to select the ADC clock source. 00: PLL clock selected as ADC clock

Field	Name	Description
		01: IHS clock selected as ADC clock 10: EHS clock selected as ADC clock
15:14	CANSEL	CAN clock source selection These bits are set and cleared by software to select the CAN clock source. 00: EHS clock selected as CAN clock 01: System clock (SYSCLK) selected as CAN clock 10: PCLK clock selected as CAN clock
13:12	USART6SEL	USART6 clock source selection This bit is set and cleared by software to select the USART6 clock source. 00: PCLK selected as USART6 clock 01: Reserved 10: IHS clock selected as USART6 clock 11: ELS clock selected as USART6 clock
11:10	USART3SEL	USART3 clock source selection This bit is set and cleared by software to select the USART3 clock source. 00: PCLK selected as USART3 clock 01: Reserved 10: IHS clock selected as USART3 clock 11: ELS clock selected as USART3 clock
9:8	USART2SEL	USART2 clock source selection This bit is set and cleared by software to select the USART2 clock source. 00: PCLK selected as USART2 clock 01: Reserved 10: IHS clock selected as USART2 clock 11: ELS clock selected as USART2 clock
7:6	USART1SEL	USART1 clock source selection

Field	Name	Description
		This bit is set and cleared by software to select the USART1 clock source. 00: PCLK selected as USART1 clock 01: Reserved 10: IHS clock selected as USART1 clock 11: ELS clock selected as USART1 clock
5:4	USART0SEL	USART0 clock source selection This bit is set and cleared by software to select the USART0 clock source. 00: PCLK selected as USART0 clock 01: Reserved 10: IHS clock selected as USART0 clock 11: ELS clock selected as USART0 clock
3:2	USART5SEL	USART5 clock source selection This bit is set and cleared by software to select the USART5 clock source. 00: PCLK selected as USART5 clock 01: Reserved 10: IHS clock selected as USART5 clock 11: ELS clock selected as USART5 clock
1:0	USART4SEL	USART4 clock source selection This bit is set and cleared by software to select the USART4 clock source. 00: PCLK selected as USART4 clock 01: Reserved 10: IHS clock selected as USART4 clock 11: ELS clock selected as USART4 clock

5.5.2.24 RCC_CFG2R

0x0098			Clock configuration register 2											RCC_CFG2R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved							TPCDIV					ADCDIV			
Type	RO							RW					RW			
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-25 Clock Configuration Register 2 Description

Field	Name	Description
31:25	Reserved	Reserved
24:20	TPCDIV	TPC divider
19:16	ADCDIV	ADC divider
15:0	Reserved	Reserved

5.5.2.25 RCC_CFG3R

0x009c			Clock configuration register 3											RCC_CFG3R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				CANTIMERDIV			TRACEDIV								
Type	RO				RW			RW								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-26 Clock Configuration Register 3 Description

Field	Name	Description
31:12	Reserved	Reserved
11:9	CANTIMERDIV	Set and cleared by software to control the division factor of the can clock (clk_can). 0xx: Not divided 100: clk_can divided by 2 101: clk_can divided by 4 110: clk_can divided by 8 111: clk_can divided by 16
8:0	TRACEDIV	TRACECLKIN div

5.5.2.26 RCC_BDCR

0x00a0			RTC domain control register											RCC_BDCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved											RSTRDY	BDRST	RTCEN	ILSRDY	ILSON
Type	RO											RO	RW	RW	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RTCSEL		ELSCSSD	ELSCSSON	ELSCTRL								ELSBYP	ELSFASTREADYEN	ELSRDY	ELSON
Type	RW		RO	RW	RW								RW	RW	RO	RW
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Table 5-27 RTC Domain Control Register Description

Field	Name	Description
31:21	Reserved	Reserved
20	RSTRDY	bd_rst ready
19	BDRST	RTC domain software reset Set and cleared by software. 0: Reset not activated 1: Reset the entire RTC domain
18	RTCEN	RTC clock enable Set and cleared by software. 0: RTC clock disabled 1: RTC clock enabled

Field	Name	Description
17	ILSRDY	ILS oscillator ready Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the ILSON bit is cleared, ILSRDY goes low after 6 external low-speed oscillator clock cycles. 0: ILS oscillator not ready 1: ILS oscillator ready
16	ILSON	ILS oscillator enable Set and cleared by software. 0: ILS oscillator off 1: ILS oscillator on
15:14	RTCSEL	RTC clock source selection Set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the RTC domain is reset or unless a failure is detected on ELS (ELSCSSD is set). The BDRST bit can be used to reset them. 00: ELS oscillator clock used as RTC clock 01: ILS oscillator clock used as RTC clock 10: EHS oscillator clock divided by 32 used as RTC clock 11: IHS oscillator clock divided by 8 used as RTC clock
13	ELSCSSD	CSS on ELS failure Detection Set by hardware to indicate when a failure has been detected by the Clock Security System on the external 32 kHz oscillator (ELS). 0: No failure detected on ELS (32 kHz oscillator) 1: Failure detected on ELS (32 kHz oscillator)
12	ELSCSSON	CSS on ELS enable Set by software to enable the Clock Security System on ELS (32 kHz oscillator). ELSCSSON must be enabled after the ELS oscillator is enabled (ELSON bit enabled) and ready (ELSRDY flag set by hardware), and after the RTCSEL bit is selected. Once enabled,

Field	Name	Description
		this bit cannot be disabled, except after a LSE failure detection (ELSCSSD =1). In that case, the software must disable the ELSCSSON bit. 0: CSS on ELS (32 kHz external oscillator) off 1: CSS on ELS (32 kHz external oscillator) on
11:4	ELSCTRL	0x00: 185 nA 0x01: 240 nA 0x03: 295 nA 0x07: 350 nA 0x0F: 405 nA 0x1F: 460 nA 0x3F: 515 nA 0x7F: 570 nA 0xFF: 625 nA
3	ELSBYP	XTAL/bypass work mode 0: XTAL mode, xa/xb connect crystal; this is the default value. 1: Bypass mode, xa/xb connect osc input
2	ELSFASTREADYEN	ELS will generated ready before setting the IB step
1	ELSRDY	ELS oscillator ready Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the ELSON bit is cleared, ELSRDY goes low after 6 external low-speed oscillator clock cycles. 0: ELS oscillator not ready 1: ELS oscillator ready
0	ELSON	ELS oscillator enable Set and cleared by software. 0: ELS oscillator off

Field	Name	Description
		1: ELS oscillator on

5.5.2.27 RCC_BDCR2

0x00a4			RTC domain control register 2											RCC_BDCR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				ILSTRIM											
Type	RO				RW											
Reset	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0

Table 5-28 RTC Domain Control Register 2 Description

Field	Name	Description
31:12	Reserved	Reserved
11:0	ILSTRIM	ILS trimming

5.5.2.28 RCC_BDCR3

0x00a8			RTC domain control register 3											RCC_BDCR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		ILSRESERVED									ILSTESTMUX				ILSTESTEN
Type	RO		RW									RW				RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-29 RTC Domain Control Register 3 Description

Field	Name	Description
31:29	Reserved	Reserved
28:21	ILSRESERVED	ILS reserved
20:17	ILSTESTMUX	DC test mux control, when ILSTESTEN = 1, this signal is valid
16	ILSTESTEN	ILS test enable control
15:0	Reserved	Reserved

5.5.2.29 RCC_BDCR4

0x00ac			RTC domain control register 4											RCC_BDCR4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ELSCSSTARGETCLKCNTEND								ELSCSSLOCALCLKCNTEND							
Type	RW								RW							
Reset	1	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Table 5-30 RTC Domain Control Register 4 Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	ELSCSSTARGETCLKCNTEND	els_css clk_els counter
7:0	ELSCSSLOCALCLKCNTEND	els_css clk_ils counter, this counter will restart from 0 when clk_els counter reach els_css_target_clk_cnt_end. If clk_ils counter reach els_css_local_clk_cnt_end, css fail detected.

5.5.2.30 RCC_BDCR5

0x00b8			RTC domain control register 5											RCC_BDCR5		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															ELSDLY EN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ELSIBSTEPDLY								ELSPREIBDLY							
Type	RW								RW							
Reset	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0

Table 5-31 RTC Domain Control Register 5 Description

Field	Name	Description
31:17	Reserved	Reserved
16	ELSDLYEN	When ELSDLYEN = 1, software can modify ELSIBSTEPDLY and ELSPREIBDLY
15:8	ELSIBSTEPDLY	Default 100ms for one step
7:0	ELSPREIBDLY	Default 1s

5.5.2.31 RCC_BDCR6

0x00bc			RTC domain control register 6											RCC_BDCR6		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved									ELSSTIB				ELSSTIBSWEN	ELSSTIBPD	ELSTESTEN
Type	RO									RW				RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0

Table 5-32 RTC Domain Control Register 6 Description

Field	Name	Description
31:7	Reserved	Reserved
6:3	ELSSTIB	After XTAL starts, decrease the ibias by logic control
2	ELSSTIBSWEN	When ELSSTIBSWEN = 1, software can modify ELSSTIB
1	ELSSTIBPD	0: Adjust ibias in start uo process 1: Stop adjust ibias in start uo process
0	ELSTESTEN	1: Enable DC test mux and clock test buffer

5.5.2.32 RCC_BDCR7

0x00c0			RTC domain control register 7											RCC_BDCR7		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															SWITCHD0EN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 5-33 RTC Domain Control Register 7 Description

Field	Name	Description
31:1	Reserved	Reserved
0	SWITCHD0EN	Switch D0 enable

Direct Memory Access (DMA)

This chapter describes the details of Direct Memory Access (DMA).

Topics: **Page**

6.1 Introduction

The Direct Memory Access (DMA) allows for high-speed data transfer between peripherals and memory or memory to memory, without requiring CPU intervention, freeing up CPU resources for other operations.

The DMA controller is built on a complex bus matrix architecture that combines a powerful dual AHB master bus architecture with a separate FIFO to optimize the system's bandwidth.

The two DMA controllers have 16 channels, with 8 channels assigned to each controller. Each channel is dedicated to managing memory access requests from one or more peripherals. Each channel can handle up to 8 requests and includes an arbiter to handle priority between the requests.

6.2 Features

- Dual AHB master bus architecture, both can access memory and peripheral.
- AHB slave programming interface supporting only 32-bit accesses.
- 8 channels for each controller, 8 requests per channel.
- Each channel has four words depth FIFO that can be used in FIFO mode or direct mode:
  - FIFO mode:

    With threshold level between 1/4, 1/2 or 3/4 of the FIFO size.
  - Direct mode:

    Each DMA request immediately initiates a transfer from/to the memory. When it is configured as direct mode, to transfer data in memory-to-peripheral mode, the DMA preloads only one data from memory to internal FIFO to ensure that data transfer is performed immediately when the peripheral triggers DMA request.
- Each channel can be configured by software to be:
  - A regular channel that supports peripheral-to-memory, memory-to-peripheral and memory-to-memory transfers.
  - A double buffer channel that supports double buffering on the memory side.
- Each of the 8 channels is connected to dedicated hardware DMA requests.
- The priority between DMA channel requests is either software programmable (4 levels including very high, high, medium, low) or hardware dependent (request 0 has higher priority than request 1, and so on).
- Each channel also supports software trigger for memory-to-memory transfers.
- Each channel request can be selected from up to 8 possible channel requests. This selection is software configurable and allows multiple peripherals to initiate DMA requests.
- The number of data items to be transferred can be managed by the DMA controller:
  - DMA flow controller: the number of data items to be transferred is software-programmable from 1 to 65535.
- Independent source and destination transfer width (byte, half-word, word): when the data widths of the source and destination are not equal, the DMA automatically packs/unpacks the necessary transfers to optimize the bandwidth. This feature is only available in FIFO mode.
- Incrementing or non-incrementing addressing for source and destination.
- Supports incremental/non-incremental burst transfers of 4, 8 or 16 beats for peripheral side and only support incremental burst transfers of 4, 8 or 16 beats for memory side. The size of the burst is software-configurable, usually equal to byte, half word or word.
- Each channel supports circular buffer working mode.

- 5 event flags (Half Transfer, Transfer complete, Transfer Error, FIFO Error, Direct Mode Error) share one single interrupt request for each channel.

6.3 Functional Description

6.3.1 General Description

Figure 6-1 shows the block diagram of Direct Memory Access (DMA) controller.

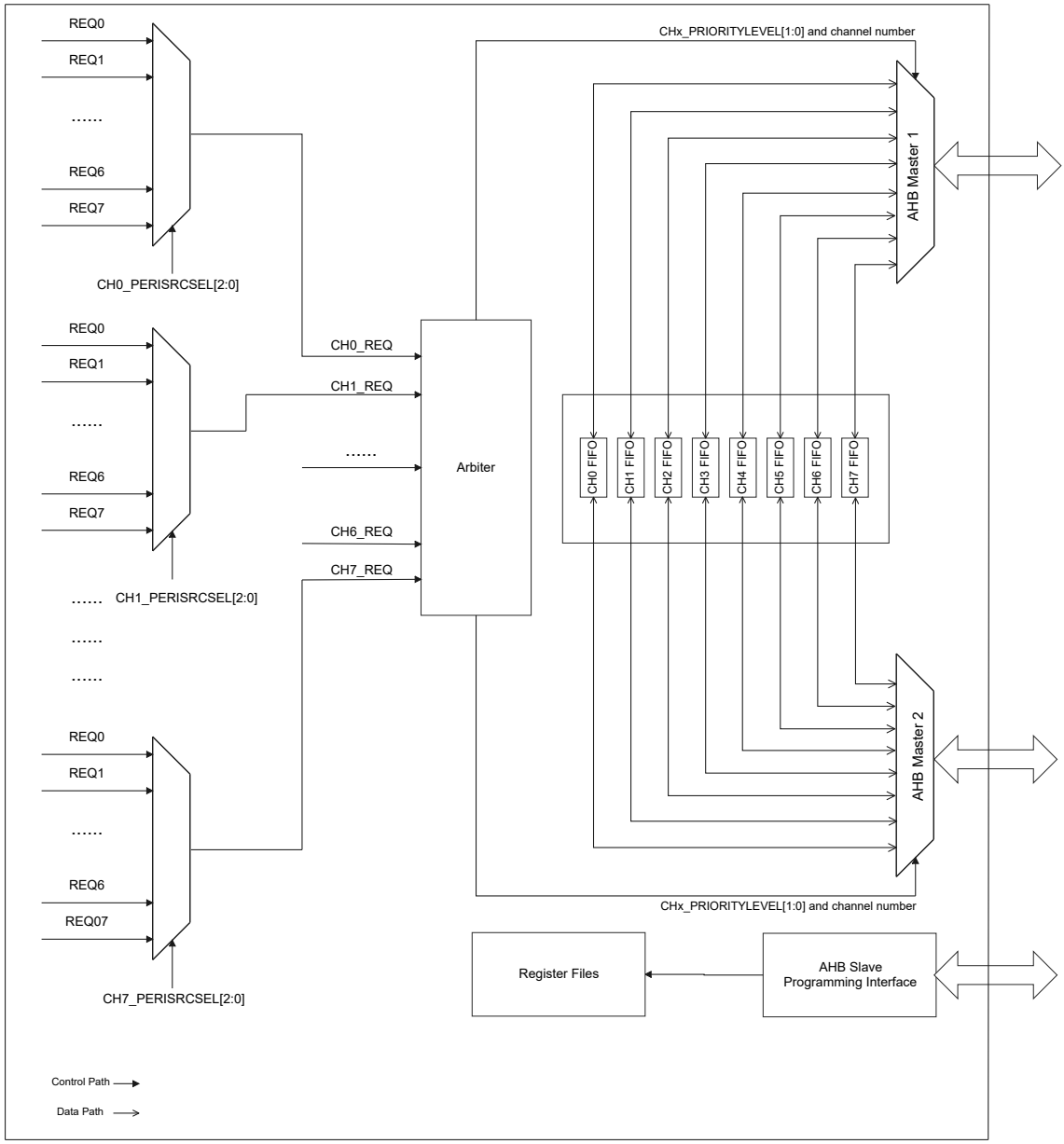


Figure 6-1 DMA Block Diagram

The DMA controller performs direct memory transfer: As an AHB master, it can take control of the AHB bus matrix to initiate AHB transactions. It can carry out the following transactions:

- Peripheral-to-memory
- Memory-to-peripherals
- Memory-to-memory

The DMA controller provides two AHB master ports: Memory port and peripheral port. Both connect the memories and peripherals.

The AHB slave port is used to program the DMA controller and it supports only 32-bit accesses.

Figure 6-2 shows the system implementation of DMA controller.

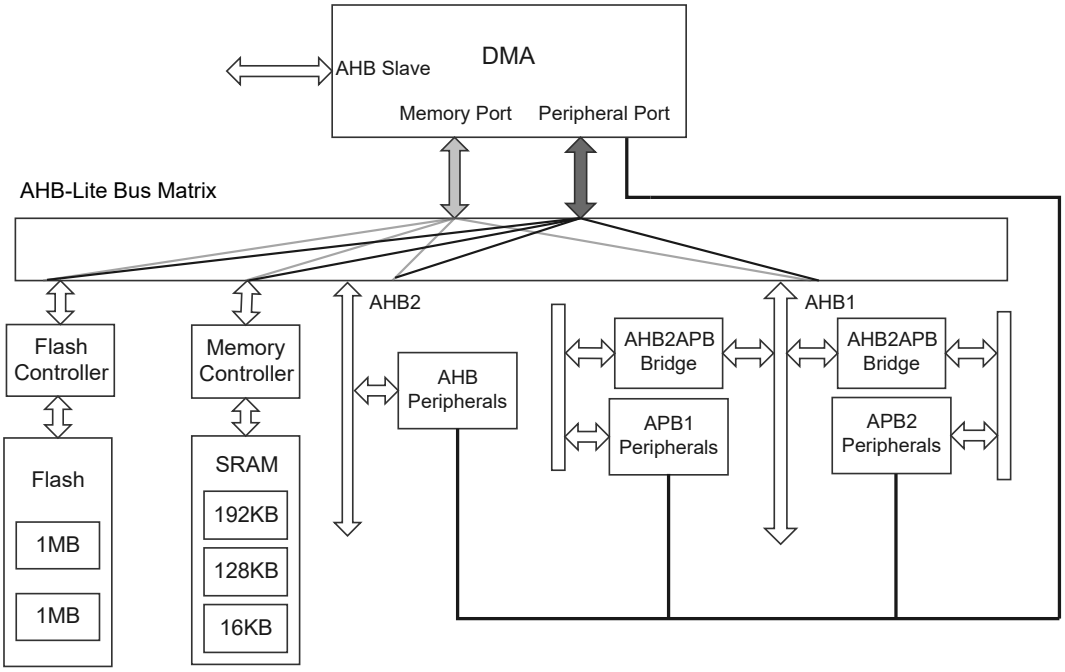


Figure 6-2 DMA Controller System Implementation

6.3.2 DMA Controller Components

6.3.2.1 DMA Channels

Each channel provides a unidirectional transfer link between a source and a destination. Each channel can be configured to perform:

- Regular type transactions: Memory-to-peripheral, peripheral-to-memory or memory-to-memory transfers.
- Double-buffer type transactions: Double buffer transfers using two memory pointers for the memory (while the DMA is reading/writing from/to a buffer, the application can write/read to/from the other buffer).

The amount of data to be transferred (up to 65535) is programmable and related to the source width of the peripheral that requests the DMA transfer connected to the peripheral AHB port. The register that contains the amount of data items to be transferred is decremented after each transaction.

6.3.2.2 DMA Requests

Each channel is associated with up to eight possible requests. The selection is controlled by the PERISRCSEL[2:0] bits in the DMA_CH_CFG register.

The eight requests from the peripherals, such as ADC, SPI, and I2C, are individually connected based on the product implementation.

For detailed DMA request mappings, please refer to the device datasheet.

6.3.2.3 DMA Arbitration

The 8 DMA channels' priorities are managed by an arbiter, which can be managed through software or hardware.

In software, the priority of each channel can be set in the DMA_CH_CFG register, with options including very high, high, medium, and low priority.

In hardware, if two requests have the same software priority level, the channel with the lower number will be given priority over the channel with the higher number (for example, channel 0 over channel 1).

6.3.2.4 First-In First-Out (FIFO)

6.3.2.4.1 FIFO Structure

The FIFO is utilized to temporarily store data originating from the source prior to transmission to the destination. Each channel possesses an independent 4-word FIFO, and the threshold level can be configured via software to be 1/4, 1/2, 3/4, or full.

In order to enable the use of the FIFO threshold level, the direct mode must be deactivated by setting the DIRECTMODEDISABLE bit in the DMA_CH_FIFO CTRL register. The structure of the FIFO varies depending on the data widths of the source and destination, as depicted in Figure 6-3.

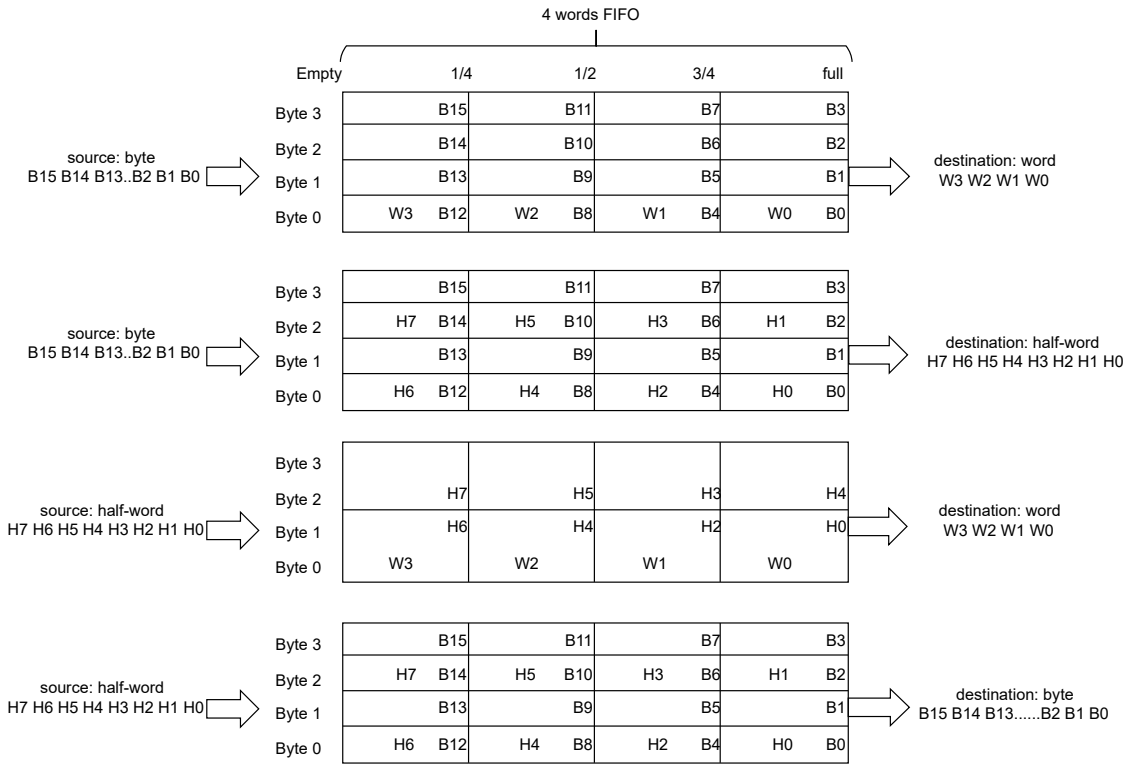


Figure 6-3 FIFO Structure

6.3.2.4.2 FIFO Threshold and Burst Configuration

When selecting the FIFO threshold (bits FIFOTH[1:0] in the DMA_CH_FIFOCTRL) and the size of the memory burst (MBURST[1:0] in the DMA_CH_CFG), it is important to ensure that the FIFO threshold corresponds to an integer number of memory burst transfers.

If not, a FIFO error (flag CHxFIFOERRINT of the DMA_INTSTS1 or DMA_INTSTS0) will be triggered when the channel is enabled, resulting in the automatic disabling of the channel. The permissible and forbidden configurations are detailed in Table 6-1.

Table 6-1 FIFO Threshold Configurations

MSIZE	FIFO Level	MBURST = INCR4	MBURST = INCR8	MBURST = INCR16
Byte	1/4	1 burst of 4 beats	Forbidden	Forbidden
	1/2	2 bursts of 4 beats	1 burst of 8 beats	
	3/4	3 bursts of 4 beats	Forbidden	
	Full	4 bursts of 4 beats	2 bursts of 8 beats	1 burst of 16 beats
Half-word	1/4	Forbidden	Forbidden	Forbidden
	1/2	1 burst of 4 beats		
	3/4	Forbidden		
	Full	2 bursts of 4 beats	1 burst of 8 beats	
Word	1/4	Forbidden	Forbidden	
	1/2			
	3/4			
	Full	1 burst of 4 beats		

In all cases, the burst size multiplied by the data size must not exceed the FIFO size (data size can be 1 (byte), 2 (half-word) or 4 (word)). Incomplete burst transfer at the end of a DMA transfer may happen if one of the following conditions occurs:

- For the AHB peripheral port configuration: the total number of data items (set in the DMA_CH_DATANUM register) is not a multiple of the burst size multiplied by the data size
- For the AHB memory port configuration: the number of remaining data items in the FIFO to be transferred to the memory is not a multiple of the burst size multiplied by the data size

In such cases, the remaining data to be transferred will be managed in single mode by the DMA, even if a burst transaction was requested during the DMA channel configuration.

NOTE: When burst transfers are requested on the peripheral AHB port and the FIFO is in use (indicated by DIRECTMODEDISABLE = 1 in the DMA_CH_CFG register), it's crucial to adhere to the following rule to prevent persistent underrun or overrun conditions, depending on the direction of the DMA channel:

If (PBURST × PERIDATASIZE) = FIFO_SIZE (4 words), setting the FIFO Threshold to 3/4 is not allowed when PERIDATASIZE is 1, 2, or 4 and PBURST is 4, 8, or 16.

This rule ensures that sufficient FIFO space will be available at any given time to accommodate the request from the peripheral.

6.3.2.4.3 FIFO Flush

The FIFO can be flushed when the channel is disabled by resetting the EN bit in the CFG register and when the channel is configured to manage peripheral-to-memory or memory-to-memory transfers: If some data are still present in the FIFO when the channel is disabled, the DMA controller continues transferring the remaining data to the destination (even though the channel is effectively disabled). When this flush is completed, the transfer complete status bit in the DMA_INTSTS0 or DMA_INTSTS1 register is set.

The remaining data counter DATANUM keeps the value, in this case, to indicate how many data items are currently available in the source memory.

Note that during the FIFO flush operation, if the number of remaining data items in the FIFO to be transferred to memory (in bytes) is less than the memory data width (for example, 2 bytes in FIFO while MEMDATASIZE is configured to word), data will be sent with the data width set in the MEMDATASIZE bit in the DMA_CH_CFG register. This means that memory will be written with an undesired value. The software may read the DMA_CH_DATANUM register to determine the memory area that contains the good data (start address and last address).

If the number of remaining data items in the FIFO is lower than a burst size (if the MBURST bits in the DMA_CH_CFG register are set to configure the channel to manage burst on the AHB memory port), single transactions will be generated to complete the FIFO flush.

6.3.2.4.4 Direct Mode

By default, the FIFO operates in direct mode (DIRECTMODEDISABLE bit in the DMA_CH_FIFOCTRL is reset) and the FIFO threshold level is not used. This mode is useful when the system requires an immediate and single transfer to or from the memory after each DMA request.

When the DMA is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.

To avoid saturating the FIFO, it is recommended to configure the corresponding channel with a high priority.

This mode is restricted to transfers where:

- The source and destination transfer widths are equal and both defined by the PERIDATASIZE[1:0] bits in DMA_CH_CFG (MEMDATASIZE[1:0] bits don't care)
- Burst transfers are not possible (PBURST[1:0] and MBURST[1:0] bits in DMA_CH_CFG are don't care)

Direct mode must not be used when implementing memory-to-memory transfers.

6.3.3 DMA Transactions

A DMA transaction comprises a particular number of data transfer sequences. The number of data items to be transferred and their width, which can be 8, 16, or 32 bits, can be programmed by the software.

Each DMA transfer sequence includes the following operations:

- Loading data from the peripheral data register or a memory location, addressed through the DMA_CH_PERIADDR or DMA_CH_MEM0ADDR register.
- Storing the loaded data to the peripheral data register or a memory location, addressed through the DMA_CH_PERIADDR or DMA_CH_MEM0ADDR register.
- Post-decrementing the DMA_CH_DATANUM register, which contains the number of transactions that are to be performed.

Following these operations, the peripheral sends a request signal to the DMA controller. The DMA controller serves requests based on their priority.

Once the DMA controller accesses the peripheral, the DMA controller sends an acknowledgment signal to the peripheral. Upon receiving the acknowledgment signal from the DMA controller, the peripheral immediately releases its request.

Once the peripheral cancels the request, the DMA controller releases an acknowledgment signal. If there are more requests, the peripheral can start the next transaction.

6.3.3.1 Transfer Modes

Both source and destination transfers can address peripherals and memories in the entire 4GB area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The direction is configured using the TRANSFERDIRECTION[1:0] bits in the CFG register and offers three possibilities: memory-to-peripheral, peripheral-to-memory or memory-to-memory transfers. Table 6-2 describes the corresponding source and destination addresses.

Table 6-2 Source and Destination Addresses

TRANSFERDIRECTION[1:0]	Direction	Source Address	Destination Address
0	Peripheral-to-memory	DMA_CH_PERIADDR	DMA_CH_MEM0ADDR
1	Memory-to-peripheral	DMA_CH_MEM0ADDR	DMA_CH_PERIADDR
10	Memory-to-memory	DMA_CH_PERIADDR	DMA_CH_MEM0ADDR
11	Reserved	-	-

When the data width (programmed in the PERIDATASIZE or MEMDATASIZE bits in the CFG register) is a half-word or a word, respectively, the peripheral or memory address written into the PERI_ADDR or MEM0_ADDR/MEM1_ADDR registers has to be aligned on a word or half-word address boundary, respectively.

6.3.3.1.1 Peripheral-to-Memory Mode

Figure 6-4 shows this mode. Upon its activation (by setting the EN bit and TRANSFERDIRECTION[1:0] bits in the DMA_CH_CFG register), each time a peripheral request occurs, the channel initiates a transfer from the source to fill the FIFO.

When the threshold level of the FIFO is reached, the contents of the FIFO are drained and stored into the destination.

The transfer stops once the DMA_CH_DATANUM register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_CH_CFG register is cleared by software.

In direct mode (when the DIRECTMODEDISABLE value in the DMA_CH_FIFOCTRL register is ‘0’), the threshold level of the FIFO is not used. After each single data transfer from the peripheral to the FIFO, the corresponding data are immediately drained and stored into the destination.

The channel has access to the AHB source or destination port only if the arbitration of the corresponding channel is won. This arbitration is performed using the priority defined for each channel using the PRIORITYLEVEL[1:0] bits in the DMA_CH_CFG register.

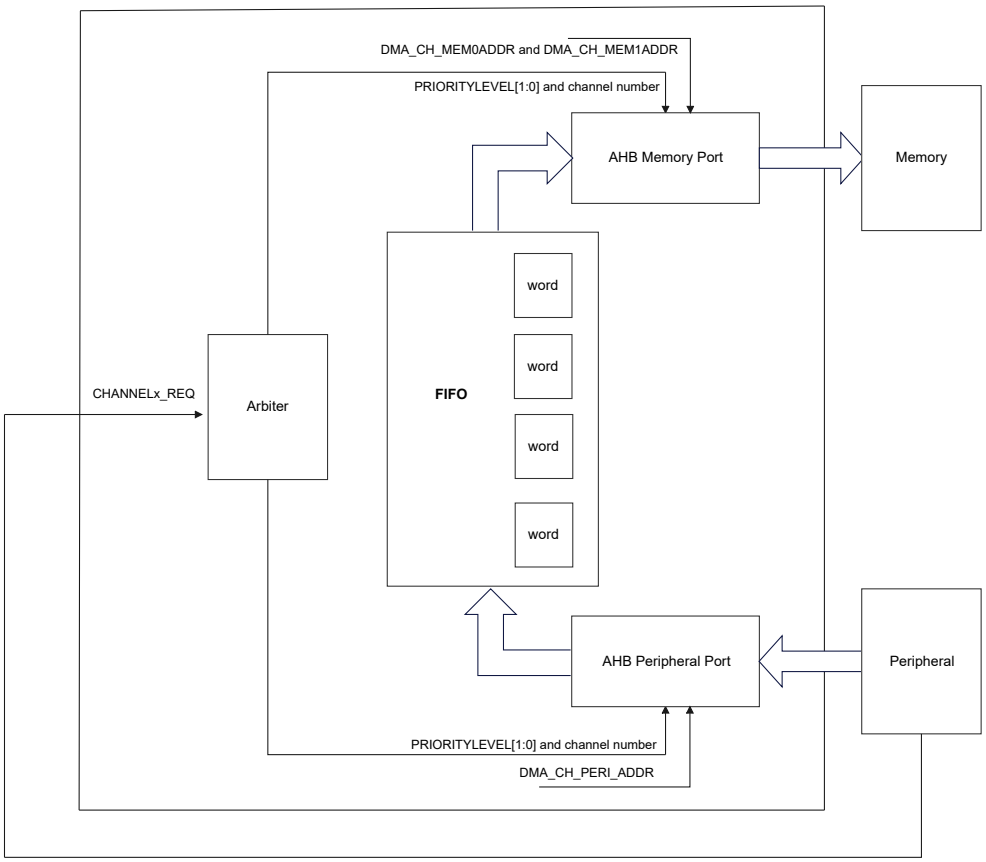


Figure 6-4 Peripheral-to-Memory Mode

6.3.3.1.2 Memory-to-Peripheral Mode

Figure 6-5 shows this mode. When this mode is enabled (by setting the bit EN and bits TRANSFERDIRECTION[1:0] in the DMA_CH_CFG register), the channel immediately initiates transfers from the source to entirely fill the FIFO.

Each time a peripheral request occurs, the contents of the FIFO are drained and stored into the destination. When the level of the FIFO is lower than or equal to the predefined threshold level, the FIFO is fully reloaded with data from the memory.

The transfer stops once the DMA_CH_DATANUM register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_CH_CFG register is cleared by software.

In direct mode (when the DIRECTMODEDISABLE value in the DMA_CH_FIFOCTRL register is '0'), the threshold level of the FIFO is not used. Once the channel is enabled, the DMA preloads the first data to transfer into an internal FIFO. As soon as the peripheral requests a data transfer, the DMA transfers the preloaded value into the configured destination. It then reloads again the empty internal FIFO with the next data to be transfer. The preloaded data size corresponds to the value of the PERIDATASIZE bitfield in the DMA_CH_CFG register.

The channel has access to the AHB source or destination port only if the arbitration of the corresponding channel is won. This arbitration is performed using the priority defined for each channel using the PRIORITYLEVEL[1:0] bits in the DMA_CH_CFG register.

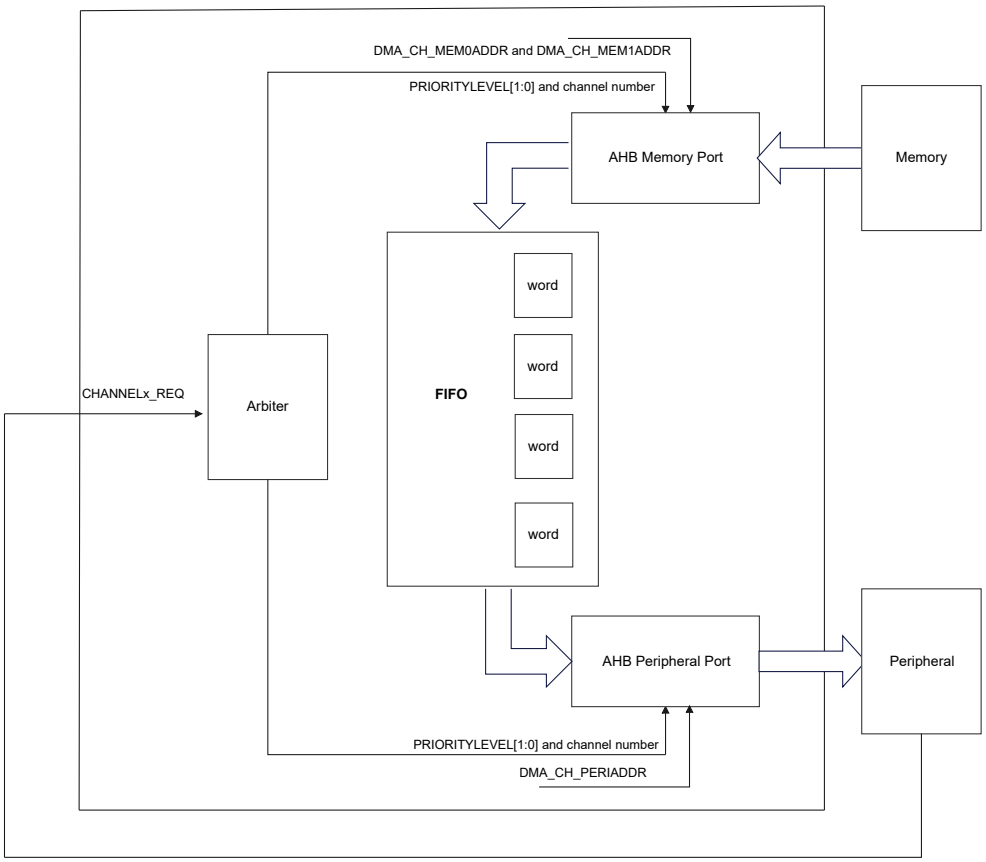


Figure 6-5 Memory-to-Peripheral Mode

6.3.3.1.3 Memory-to-Memory Mode

The DMA requests can also work without being triggered by a request from a peripheral. This is the memory-to-memory mode, described in Figure 6-6.

When the channel is enabled by setting the Enable bit (EN) in the DMA_CH_CFG register, the channel immediately starts to fill the FIFO up to the threshold level. When the threshold level is reached, the FIFO contents are drained and stored into the destination.

The transfer stops once the DMA_CH_DATANUM register reaches zero or when the EN bit in the DMA_CH_CFG register is cleared by software.

The channel has access to the AHB source or destination port only if the arbitration of the corresponding channel is won. This arbitration is performed using the priority defined for each channel using the PRIORITYLEVEL[1:0] bits in the DMA_CH_CFG register.

NOTE: When memory-to-memory mode is used, the Circular and direct modes are not allowed.

Both DAM0 and DMA1 controllers support memory-to-memory transfers.

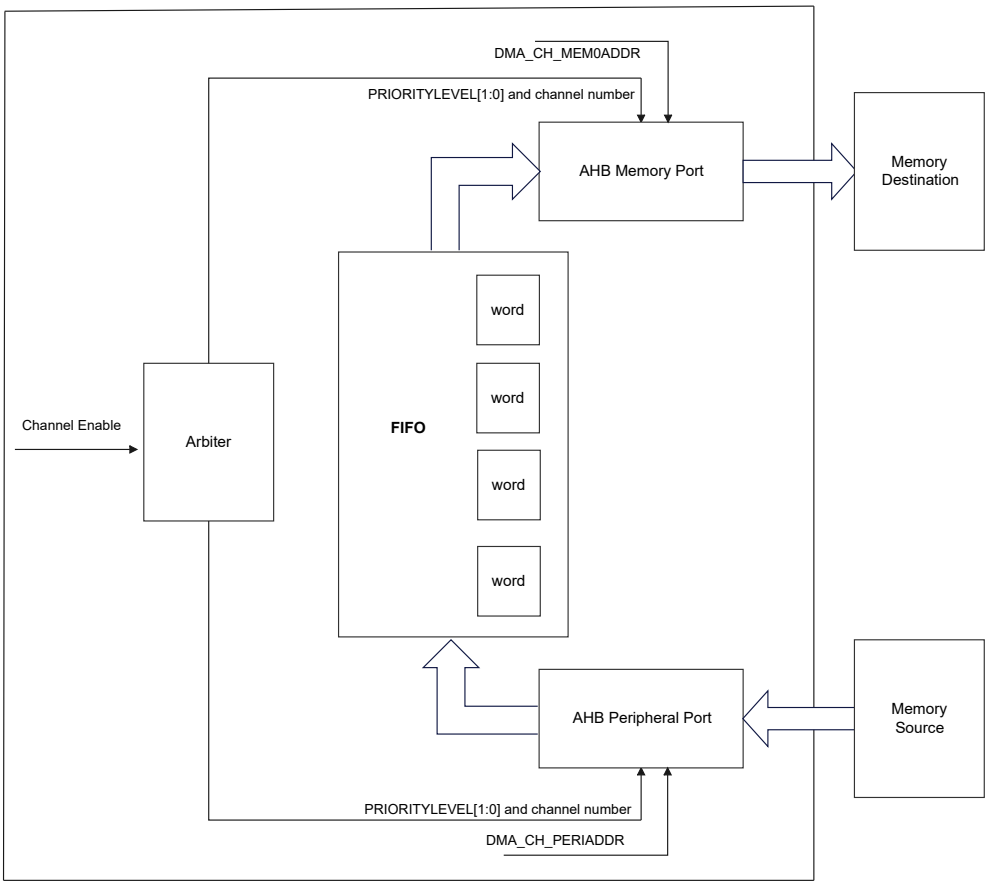


Figure 6-6 Memory-to-Memory Mode

6.3.3.2 Single and Burst Transfers

The DMA controller can generate single transfers or incremental burst transfers of 4, 8 or 16 beats.

The size of the burst is configured by software independently for the two AHB ports by using the MBURST[1:0] and PBURST[1:0] bits in the DMA_CH_CFG register.

The burst size indicates the number of beats in the burst, not the number of bytes transferred.

To ensure data coherence, each group of transfers that form a burst are indivisible: AHB transfers are locked and the arbiter of the AHB bus matrix does not disturb the DMA master during the sequence of the burst transfer.

Depending on the single or burst configuration, each DMA request initiates a different number of transfers on the AHB peripheral port:

- When the AHB peripheral port is configured for single transfers, each DMA request generates a data transfer of a byte, half-word or word depending on the PERIDATASIZE[1:0] bits in the DMA_CH_CFG register
- When the AHB peripheral port is configured for burst transfers, each DMA request generates 4,8 or 16 beats of byte, half word or word transfers depending on the PBURST[1:0] and PERIDATASIZE[1:0] bits in the DMA_CH_CFG register.

The same as above has to be considered for the AHB memory port considering the MBURST and MEMDATASIZE bits.

In direct mode, the channel can only generate single transfers and the MBURST[1:0] and PBURST[1:0] bits are forced by hardware.

The address pointers (DMA_CH_PERIADDR or DMA_CH_MEM0ADDR registers) must be chosen to ensure that all transfers within a burst block are aligned on the address boundary equal to the size of the transfer.

1KB address boundary could be crossed by burst block transfer, the burst on 1K boundary will switch to single mode automatically.

6.3.3.3 DMA Transfer Completion

Various events can trigger the termination of a transfer by enabling the CHxFULLTRANSFERINT bit in the DMA_CH_INTSTS0 or DMA_CH_INTSTS1 register. These events include:

- In DMA flow controller mode, the DATANUM counter reaches zero in the memory-to-peripheral mode.
- The channel is disabled before the transfer is complete by clearing the EN bit in the CFG register. In peripheral-to-memory or memory-to-memory transfers, all remaining data is flushed from the FIFO into the memory.

NOTE: The transfer completion depends on the remaining data in FIFO being transferred into memory only in the case of peripheral-to-memory mode. This condition does not apply in memory-to-peripheral mode.

If the channel is configured in noncircular mode, the DMA will stop after the transfer is complete (when the number of data to be transferred reaches zero). In this case, no DMA request will be served unless the software reprograms and re-enables the channel by setting the EN bit in the DMA_CH_CFG register.

6.3.3.4 Flow Controller

The entity that controls the number of data to be transferred is known as the flow controller. This flow controller is configured independently for each channel using the TRANSFERCONTROLLERSEL bit in the DMA_CH_CFG register.

The flow controller can be the DMA controller. In this case, the number of data items to be transferred is programmed by software into the DATANUM register before the DMA channel is enabled.

6.3.3.5 DMA Transfer Suspension

At any time, a DMA transfer can be suspended to be restarted later or to be permanently disabled before the end of the DMA transfer:

1. **Suspending the transfer**:

   In this case, the channel suspends the transfer before the remaining number of data items to be transferred reaches 0, as indicated by the DMA_CH_DATANUM register.

   The aim is to restart the transfer later by re-enabling the channel. To restart from the point where the transfer was stopped, the software needs to perform the following steps:

   a. Disable the channel by clearing the EN bit in the DMA_CH_CFG register, and ensure that the EN bit is set to '0' to confirm the channel has been disabled.
   b. Read the DMA_CH_DATANUM register to determine the number of data items that have already been transferred before the channel was disabled.
   c. Update the peripheral and/or memory addresses to adjust the address pointers accordingly.
   d. Update the DMA_CH_DATANUM register with the remaining number of data items to be transferred, using the value previously read when the channel was disabled.
   e. Finally, re-enable the channel to restart the transfer from the point where it was suspended.
2. **Disabling the transfer**:

In this case, the channel disables the transfer without the possibility of restarting it later. To disable the channel, the EN bit in the DMA_CH_CFG register should be cleared. It may take some time for the channel to be disabled, as any ongoing transfer will be completed first.

The CHxFULLTRANSFERINT flag in either the DMA_CH_INTSTS0 or DMA_CH_INTSTS1 register is set to indicate the end of the transfer. The EN bit in the DMA_CH_CFG register is now set to '0' to confirm the interruption of the channel. The DMA_CH_DATANUM register contains the number of remaining data items at the moment the channel was stopped, allowing the software to determine how many data items were transferred before the interruption.

NOTE: A transfer complete interrupt flag (CHxFULLTRANSFERINT in DMA_CH_INTSTS0 or DMA_CH_INTSTS1) is set to indicate the end of the transfer caused by a channel interruption.

6.3.3.6 Pointer Incrementation

Peripheral and memory pointers can optionally be automatically post-incremented or kept constant after each transfer depending on the PERIINCEN and MEMINCEN bits in the DMA_CH_CFG register.

Disabling the increment mode is useful when the peripheral source or destination data are accessed through a single register.

If the increment mode is enabled, the address of the next transfer will be the address of the previous one incremented by 1 (for bytes), 2 (for half-words) or 4 (for words) depending on the data width programmed in the PERIDATASIZE or MEMDATASIZE bits in the DMA_CH_CFG register.

To optimize the packing operation, it is possible to fix the increment offset size for the peripheral address whatever the size of the data transferred on the AHB peripheral port. The PERIINCOFFSETSIZE bit in the DMA_CH_CFG register is used to align the increment offset size with the data size on the peripheral AHB port, or on a 32-bit address (the address is then incremented by 4). The PERIINCOFFSETSIZE bit has an impact on the AHB peripheral port only.

Table 6-3 PERIINCOFFSETSIZE Impact on Increment

PERIINCOFFSETSIZE	Increment
0	PERIDATASIZE
1	4 (32 bits)

6.3.4 Special Working Mode

6.3.4.1 Circular Mode

The circular mode is available to handle circular buffers and continuous data flows, such as ADC scan mode. This feature can be enabled using the CIRCULARMODE bit in the DMA_CH_CFG register.

When the circular mode is activated, the number of data items to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

NOTE: In the circular mode, it is important to follow a specific rule when configuring a burst mode for memory. The rule is as follows:

DMA_CH_DATANUM = Multiple of ((Mburst beat) × (Msize)/(Psize))

Where:

- (Mburst beat) = 4, 8 or 16 (depending on the MBURST bits in the CFG register)
- ((Msize)/(Psize)) = 1, 2, 4, 1/2 or 1/4 (Msize and Psize represent the MEMDATASIZE and PERIDATASIZE bits in the CFG register. They are byte dependent.)
- DMA_CH_DATANUM = Number of data items to transfer on the AHB peripheral port

For example, Mburst beat = 8 (INCR8), MEMDATASIZE = '00' (byte) and PERIDATASIZE = '01' (half-word), in this case: DATANUM must be a multiple of (8 × 1/2 = 4).

It is important to note that if this formula is not followed, it can lead to unpredictable behavior of the DMA and may compromise data integrity. Additionally, DMA_CH_DATANUMR must also be a multiple of the peripheral burst size multiplied by the peripheral data size to avoid any issues with DMA behavior.

6.3.4.2 Double Buffer Mode

This mode is available for all the DMA0 and DMA1 channels.

The double buffer mode is enabled by setting the MEMSWITCHMODE bit in the DMA_CH_CFG register.

A double buffer channel works as a regular (single buffer) channel with the difference that it has two memory pointers. When the double buffer mode is enabled, the circular mode is automatically enabled (CIRCULARMODE bit in DMA_CH_CFG doesn't care), and at each end of the transaction, the memory pointers are swapped.

In this mode, the DMA controller swaps from one memory target to another at each end of the transaction. This allows the software to process one memory area while the second memory area is being filled/used by the DMA transfer. The double buffer channel can work in both directions (the memory can be either the source or the destination).

NOTE: In double buffer mode, it is possible to update the base address for the AHB memory port on-the-fly (MEM0_ADDR or MEM1_ADDR) when the channel is enabled by respecting the following conditions:

- When the CURRENTMEM bit is 0 in the DMA_CH_CFG register, writing to the DMA_CH_MEM0ADDR register is not allowed. Attempting to write to this register while CURRENTMEM = 0 will set an error flag (CHxTRANSFERERRINT), and the channel is automatically disabled. However, writing to the DMA_CH_MEM1ADDR register is allowed.
- When the CURRENTMEM bit is 1 in the DMA_CH_CFG register, writing to the MEM1_ADDR register is not allowed. Attempting to write to this register while CURRENTMEM = 1 will set an error flag (CHxTRANSFERERRINT), and the channel is automatically disabled. However, writing to the DMA_CH_MEM0ADDR register is allowed.

To avoid any error condition, it is advised to change the base address as soon as the CHxFULLTRANSFERINT flag is asserted. At this point, the targeted memory should have changed from memory 0 to 1 (or from 1 to 0) based on the value of CURRENTMEM in the DMA_CH_CFG register and the conditions mentioned above.

For all the other modes (except the double buffer mode), the memory address registers are write-protected once the channel is enabled.

Table 6-4 Source and Destination Address Registers in Double Buffer Code (MEMSWITCHMODE = 1)

TRANSFERDIRECTION[1:0]	Direction	Source Address	Destination Address
01	Memory-to-peripheral	MEM0_ADDR and MEM1_ADDR	PERI_ADDR
10	Not allowed (1)		
11	Reserved	-	-

(1) When the double buffer mode is enabled, the circular mode is automatically enabled. Since the memory-to-memory mode is not compatible with the circular mode, when the double buffer mode is enabled, it is not allowed to configure the memory-to-memory mode.

6.3.5 DMA Configuration

6.3.5.1 Possible Configurations

Table 6-5 summarizes the different possible DMA configurations.

Table 6-5 DMA Configurations

DMA Transfer Mode	Source	Destination	Flow Controller	Circular Mode	Transfer Type	Direct Mode	Double Buffer Mode
Peripheral-to-memory	AHB peripheral port	AHB memory port	DMA	Possible	Single	Possible	Possible
					Burst	Forbidden	
			Peripheral	Forbidden	Single	Possible	Forbidden
					Burst	Forbidden	
Memory-to-peripheral	AHB memory port	AHB peripheral port	DMA	Possible	Single	Possible	Possible
					Burst	Forbidden	
			Peripheral	Forbidden	Single	Possible	Forbidden
					Burst	Forbidden	
Memory-to-memory	AHB peripheral port	AHB memory port	DMA only	Forbidden	Single	Forbidden	Forbidden
					Burst		

6.3.5.2 Data Width, Packing/Unpacking and Endianess

The transfer channel must be enabled after programming the number of data items to be transferred into the DATANUM bit, except when the flow controller is the peripheral and TRANSFERCONTROLLERSEL bit is set in DMA_CH_CFG.

When using the internal FIFO, the data widths of the source and destination data are programmable through the PERIDATASIZE and MEMDATASIZE bits in the DMA_CH_CFG register (can be 8, 16 or 32 bits).

When PERIDATASIZE and MEMDATASIZE are not equal:

- The data width of the number of data items to transfer, configured in the DMA_CH_DATANUM register, is equal to the width of the peripheral bus (configured by the PERIDATASIZE bits in the DMA_CH_CFG register). For example, in case of peripheral-to-memory, memory-to-peripheral or memory-to-memory transfers and if the PERIDATASIZE[1:0] bits are configured for half-word, the number of bytes to be transferred is equal to 2 × DATANUM.
- The DMA controller only supports little-endian addressing for both source and destination, as described in Table 6-6.

The process of packing and unpacking may pose a risk of data corruption if interrupted before completion. To ensure data integrity, the channel can be configured to perform burst transfers. In this scenario, each set of transfers within a burst cannot be divided (refer to Single and Burst Transfers).

When in direct mode (with DIRECTMODEDISABLE set to 0 in the DMA_CH_FIFOCTRL register), data packing and unpacking are not feasible. In such cases, the source and destination transfer data widths must be identical and are determined by the PERIDATASIZE bits in the DMA_CH_CFG, while the MEMDATASIZE bits are irrelevant.

Table 6-6 Packing/unpacking & Endian Behavior (Bit PERIINCEN = MEMINCEN = 1)

AHB Memory Port Width	AHB Peripheral Port Width	DATANUM (1)	Memory Transfer Number	Memory Port Address/Byte Lane	Peripheral Transfer Number	Peripheral Port (2) Address/Byte Lane	
						PERIINCOFFSETSIZE = 1	PERIINCOFFSETSIZE = 0
8	8	4	1	0x0 / B0[7:0]	1	0x0 / B0[7:0]	0x0 / B0[7:0]
			2	0x1 / B1[7:0]	2	0x4 / B1[7:0]	0x1 / B1[7:0]
			3	0x2 / B2[7:0]	3	0x8 / B2[7:0]	0x2 / B2[7:0]
			4	0x3 / B3[7:0]	4	0xC / B3[7:0]	0x3 / B3[7:0]
8	16	2	1	0x0 / B0[7:0]	1	0x0 / B1|B0[15:0]	0x0 / B1|B0[15:0]
			2	0x1 / B1[7:0]			
			3	0x2 / B2[7:0]	2	0x4 / B3|B2[15:0]	0x2 / B3|B2[15:0]

AHB Memory Port Width	AHB Peripheral Port Width	DATANUM (1)	Memory Transfer Number	Memory Port Address/Byte Lane	Peripheral Transfer Number	Peripheral Port (2) Address/Byte Lane	
						PERIINCOFFSETSIZE = 1	PERIINCOFFSETSIZE = 0
			4	0x3 / B3[7:0]			
8	32	1	1	0x0 / B0[7:0]	1	0x0 / B3|B2|B1| B0[31:0]	0x0 / B3|B2|B1| B0[31:0]
			2	0x1 / B1[7:0]			
			3	0x2 / B2[7:0]			
			4	0x3 / B3[7:0]			
16	8	4	2	0x0 / B1|B0[15:0]	1	0x0 / B0[7:0]	0x0 / B0[7:0]
					2	0x4 / B1[7:0]	0x1 / B1[7:0]
				0x2 / B3|B2[15:0]	3	0x8 / B2[7:0]	0x2 / B2[7:0]
					4	0xC / B3[7:0]	0x3 / B3[7:0]
16	16	2	1	0x0 / B1|B0[15:0]	1	0x0 / B1|B0[15:0]	0x0 / B1|B0[15:0]
			2	0x2 / B1|B0[15:0]	2	0x4 / B3|B2[15:0]	0x2 / B3|B2[15:0]
16	32	1	1	0x0 / B1|B0[15:0]	1	0x0 / B3|B2|B1| B0[31:0]	0x0 / B3|B2|B1| B0[31:0]
			2	0x2 / B3|B2[15:0]			
32	8	4	1	0x0 / B3|B2|B1| B0[31:0]	1	0x0 / B0[7:0]	0x0 / B0[7:0]
					2	0x4 / B1[7:0]	0x1 / B1[7:0]
					3	0x8 / B2[7:0]	0x2 / B2[7:0]
					4	0xC / B3[7:0]	0x3 / B3[7:0]
32	16	2	1	0x0 /B3|B2|B1| B0[31:0]	1	0x0 / B1|B0[15:0]	0x0 / B1|B0[15:0]
					2	0x4 / B3|B2[15:0]	0x2 / B3|B2[15:0]
32	32	1	1	0x0 /B3|B2|B1|B0 [31:0]	1	0x0 /B3|B2|B1|B0 [31:0]	0x0 / B3|B2|B1| B0[31:0]

(1): Number of data items to transfer.

(2): Peripheral port can act as either the source or the destination. It can also function as the memory source during memory-to-memory transfer.

PERIDATASIZE, MEMDATASIZE, and DATANUM[15:0] should be configured to prevent incomplete transfers. This situation can arise when the peripheral port's data width (PERIDATASIZE bits) is smaller than the memory port's data width (MEMDATASIZE bits). This requirement is outlined in Table 6-7.

Table 6-7 Restriction on DATANUM versus PERIDATASIZE and MEMDATASIZE

PERIDATASIZE[1:0]	MEMDATASIZE[1:0]	DATANUM[15:0]
00 (8-bit)	01 (16-bit)	Must be a multiple of 2
00 (8-bit)	10 (32-bit)	Must be a multiple of 4
01 (16-bit)	10 (32-bit)	Must be a multiple of 2

6.3.5.3 Channel Configuration

To configure DMA channel x (where x represents the channel number), follow these steps:

1. If the channel is enabled, disable it by resetting the EN bit in the DMA_CH_CFG register. Then, read this bit to confirm that there is no ongoing channel operation.

   Writing this bit to 0 does not immediately take effect; it is actually written to 0 once all the current transfers have finished. When the EN bit is read as 0, it indicates that the channel is ready to be configured. Therefore, it is necessary to wait for the EN bit to be cleared before starting any channel configuration. Before re-enabling the channel, make sure to clear all the channel dedicated bits that were set in the status register (DMA_CH_INTSTS0 and DMA_CH_INTSTS1) from the previous data block DMA transfer.
2. Specify the address of the peripheral port register in the DMA_CH_PERIADDR register.

   This will facilitate the transfer of data to or from the peripheral port subsequent to the peripheral event.
3. Set the memory address in the DMA_CH_MEM0ADDR register (and in the case of double buffer mode, also in the DMA_CH_MEM1ADDR register).

   The data will be written to or read from this memory after the peripheral event.
4. Configure the total number of data items to be transferred in the DMA_CH_DATANUM register. The value will decrement after each peripheral event or burst beat.
5. Select the DMA request using PERISRCSEL[2:0] in the DMA_CH_CFG register.
6. If the peripheral is intended to act as the flow controller and supports this functionality, then enable the TRANSFERCONTROLLERSEL bit in the DMA_CH_CFG register.
7. Configure the channel priority using the PRIORITYLEVEL[1:0] bits in the DMA_CH_CFG register.
8. Configure the usage of FIFO (enable or disable, transmission and reception threshold).
9. Configure the data transfer direction, peripheral and memory incremented/fixed mode, single or burst transactions, peripheral and memory data widths, circular mode, double buffer mode and interrupts after half and/or full transfer, and/or errors in the DMA_CH_CFG register.
10. Enable the channel by setting the EN bit in the DMA_CH_CFG register. Once the channel is activated, it can handle any DMA request from the peripheral connected to it.

When half of the data has been transferred to the AHB destination port, the CHxHALFTRANSFERINT flag is set, and an interrupt is generated if the HALFTRANSFERINTEN bit is enabled. Upon completion of the transfer, the CHxFULLTRANSFERINT flag is set, and an interrupt is generated if the FULLTRANSFERINTEN bit is enabled.

NOTE: To disable a peripheral connected to a DMA channel request, it is necessary to first deactivate the DMA channel that the peripheral is connected to. Then, wait for the EN bit to become 0. Only after these steps, the peripheral can be safely switched off.

6.3.6 Error Management

The DMA controller is capable of detecting the following errors:

- **Transfer error**: The transfer error interrupt flag (CHxTRANSFERERRINT) is set when:
  - A bus error occurs during a DMA read or a write access.
  - Software requests a write access on a memory address register in double buffer mode while the channel is enabled and the current target memory is affected by the write into the memory address register.
- **FIFO error**: The FIFO error interrupt flag (CHxFIFOERRINT) is set if:
  - A FIFO underrun condition is detected.

  - A FIFO overrun condition is detected (no detection in memory-to-memory mode because requests and transfers are internally managed by the DMA).
  - The channel is enabled while the FIFO threshold level is incompatible with the size of the memory burst.
- **Direct mode error**: The direct mode error interrupt flag (CHxDIRECTMODEERRINT) can only be set in the peripheral-to-memory mode while operating in direct mode and when the MEMINCEN bit in the DMA_CH_CFG register is cleared. This flag is set when a DMA request occurs while the previous data have not yet been fully transferred into the memory (because the memory bus was not granted). In this case, the flag indicates that two data items were transferred successively to the same destination address, which could be problematic if the destination cannot handle this situation.

  In direct mode, the FIFO error flag can also be set under the following conditions:

  - In the peripheral-to-memory mode, the FIFO can be saturated (overrun) if the memory bus is not granted for several peripheral requests.
  - In the memory-to-peripheral mode, an underrun condition may occur if the memory bus has not been granted before a peripheral request occurs.

If the CHxTRANSFERERRINT or the CHxFIFOERRINT flag is set due to incompatibility between burst size and FIFO threshold level, the faulty channel is automatically disabled through hardware clear of its EN bit in the corresponding DMA_CH_CFG.

If the CHxDIRECTMODEERRINT or CHxFIFOERRINT flag is set due to an overrun or underrun condition, the faulty channel is not automatically disabled, and it is up to the software to decide whether or not to disable the channel by resetting the DMA_CH_CFG register. This is because no data loss occurs when these types of errors occur

When the channel's error interrupt flag (TRANSFERERRINTEN, FIFOERRINTEN, DIRECTMODEERRINTEN) in the DMA_CH_INTSTS0 or DMA_CH_INTSTS1 register is set, an interrupt is generated if the corresponding interrupt enable bit (TRANSFERERRINTEN, FIFOERRINTEN, DIRECTMODEERRINTEN) in the DMA_CH_CFG or DMA_CH_FIFOCTRL register is set.

NOTE: In the event of a FIFO overrun or underrun, the data will not be lost as the peripheral request is not acknowledged by the channel until the overrun or underrun condition is cleared. However, if the acknowledgment takes too long, the peripheral may detect an overrun or underrun condition in its internal buffer, resulting in potential data loss.

6.4 Interrupts

For each DMA channel, an interrupt can be produced on the following events:

- Half-transfer reached
- Transfer complete
- Transfer error
- FIFO error (overrun, underrun or FIFO level error)
- Direct mode error

Separate interrupt enable control bits are available for flexibility.

6.5 Registers

6.5.1 Register Address Map

The DMA registers have to be accessed by words (32 bits).

Offset	Register Name	Register Description
0x0000	DMA_INTSTS0	DMA interrupt status register 0
0x0004	DMA_INTSTS1	DMA interrupt status register 1
0x0008	DMA_INTCLR0	DMA interrupt clear register 0
0x000c	DMA_INTCLR1	DMA interrupt clear register 1

The following registers are defined for channels (8 channels in each DMA controller).

Offset	Register Name	Register Description
0x0000	DMA_CH_CFG	DMA channel configuration register
0x0004	DMA_CH_DATANUM	DMA channel data number register
0x0008	DMA_CH_PERIADDR	Peripheral base address register
0x000c	DMA_CH_MEM0ADDR	MEM0 base address register
0x0010	DMA_CH_MEM1ADDR	MEM1 base address register
0x0014	DMA_CH_FIFOCTRL	FIFO control register

6.5.2 Register Field Details

6.5.2.1 DMA_INTSTS0

0x0000			DMA interrupt status register 0											DMA_INTSTS0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				CH3FULLTRANSFERINT	CH3HALFTRANSFERINT	CH3TRANSFERERRINT	CH3DIRECTMODEERRINT	Reserved	CH3FIFOERRINT	CH2FULLTRANSFERINT	CH2HALFTRANSFERINT	CH2TRANSFERERRINT	CH2DIRECTMODEERRINT	Reserved	CH2FIFOERRINT
Type	RO				RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				CH1FULLTRANSFERINT	CH1HALFTRANSFERINT	CH1TRANSFERERRINT	CH1DIRECTMODEERRINT	Reserved	CH1FIFOERRINT	CH0FULLTRANSFERINT	CH0HALFTRANSFERINT	CH0TRANSFERERRINT	CH0DIRECTMODEERRINT	Reserved	CH0FIFOERRINT
Type	RO				RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-8 DMA Interrupt Status Register 0 Description

Field	Name	Description
31:28	Reserved	Reserved
27	CH3FULLTRANSFERINT	Full transfer interrupt flag

Field	Name	Description
26	CH3HALFTRANSFERINT	Half transfer interrupt flag
25	CH3TRANSFERERRINT	Transfer error interrupt flag
24	CH3DIRECTMODEERRINT	Direct mode error interrupt flag
23	Reserved	Reserved
22	CH3FIFOERRINT	FIFO error interrupt flag
21	CH2FULLTRANSFERINT	Full transfer interrupt flag
20	CH2HALFTRANSFERINT	Half transfer interrupt flag
19	CH2TRANSFERERRINT	Transfer error interrupt flag
18	CH2DIRECTMODEERRINT	Direct mode error interrupt flag
17	Reserved	Reserved
16	CH2FIFOERRINT	FIFO error interrupt flag
15:12	Reserved	Reserved
11	CH1FULLTRANSFERINT	Full Transfer interrupt flag
10	CH1HALFTRANSFERINT	Half Transfer interrupt flag
9	CH1TRANSFERERRINT	Transfer Error interrupt flag
8	CH1DIRECTMODEERRINT	Direct mode Error interrupt flag
7	Reserved	Reserved
6	CH1FIFOERRINT	FIFO Error interrupt flag
5	CH0FULLTRANSFERINT	Full Transfer interrupt flag
4	CH0HALFTRANSFERINT	Half Transfer interrupt flag
3	CH0TRANSFERERRINT	Transfer Error interrupt flag

Field	Name	Description
2	CH0DIRECTMODEERRINT	Direct mode Error interrupt flag
1	Reserved	Reserved
0	CH0FIFOERRINT	FIFO Error interrupt flag

6.5.2.2 DMA_INTSTS1

0x0004	DMA interrupt status register 1													DMA_INTSTS1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				CH7FULLTRANSFERINT	CH7HALFTRANSFERINT	CH7TRANSFERERRINT	CH7DIRECTMODEERRINT	Reserved	CH7FIFOERRINT	CH6FULLTRANSFERINT	CH6HALFTRANSFERINT	CH6TRANSFERERRINT	CH6DIRECTMODEERRINT	Reserved	CH6FIFOERRINT
Type	RO				RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				CH5FULLTRANSFERINT	CH5HALFTRANSFEERINT	CH5TRANSFERERRINT	CH5DIRECTMODEERRINT	Reserved	CH5FIFOERRINT	CH4FULLTRANSFERINT	CH4HALFTRANSFEERINT	CH4TRANSFERERRINT	CH4DIRECTMODEERRINT	Reserved	CH4FIFOERRINT
Type	RO				RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-9 DMA Interrupt Status Register 1 Description

Field	Name	Description
31:28	Reserved	Reserved
27	CH7FULLTRANSFERINT	Full transfer interrupt flag
26	CH7HALFTRANSFERINT	Half transfer interrupt flag

Field	Name	Description
25	CH7TRANSFERERRINT	Transfer error interrupt flag
24	CH7DIRECTMODEERRINT	Direct mode error interrupt flag
23	Reserved	Reserved
22	CH7FIFOERRINT	FIFO error interrupt flag
21	CH6FULLTRANSFERINT	Full transfer interrupt flag
20	CH6HALFTRANSFERINT	Half transfer interrupt flag
19	CH6TRANSFERERRINT	Transfer error interrupt flag
18	CH6DIRECTMODEERRINT	Direct mode error interrupt flag
17	Reserved	Reserved
16	CH6FIFOERRINT	FIFO error interrupt flag
15:12	Reserved	Reserved
11	CH5FULLTRANSFERINT	Full transfer interrupt flag
10	CH5HALFTRANSFERINT	Half transfer interrupt flag
9	CH5TRANSFERERRINT	Transfer error interrupt flag
8	CH5DIRECTMODEERRINT	Direct mode error interrupt flag
7	Reserved	Reserved
6	CH5FIFOERRINT	FIFO error interrupt flag
5	CH4FULLTRANSFERINT	Full transfer interrupt flag
4	CH4HALFTRANSFERINT	Half transfer interrupt flag
3	CH4TRANSFERERRINT	Transfer error interrupt flag
2	CH4DIRECTMODEERRINT	Direct mode error interrupt flag

Field	Name	Description
1	Reserved	Reserved
0	CH4FIFOERRINT	FIFO error interrupt flag

6.5.2.3 DMA_INTCLR0

0x0008	DMA interrupt clear register 0													DMA_INTCLR0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				CH3FULLTRANSFERINTCLR	CH3HALFTRANSFERINTCLR	CH3TRANSFERERRINTCLR	CH3DIRECTMODEERRINTCLR	Reserved	CH3FIFOERRINTCLR	CH2FULLTRANSFERINTCLR	CH2HALFTRANSFERINTCLR	CH2TRANSFERERRINTCLR	CH2DIRECTMODEERRINTCLR	Reserved	CH2FIFOERRINTCLR
Type	RO				WC	WC	WC	WC	RO	WC	WC	WC	WC	WC	RO	WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				CH1FULLTRANSFERINTCLR	CH1HALFTRANSFERINTCLR	CH1TRANSFERERRINTCLR	CH1DIRECTMODEERRINTCLR	Reserved	CH1FIFOERRINTCLR	CH0FULLTRANSFERINTCLR	CH0HALFTRANSFERINTCLR	CH0TRANSFERERRINTCLR	CH0DIRECTMODEERRINTCLR	Reserved	CH0FIFOERRINTCLR
Type	RO				WC	WC	WC	WC	RO	WC	WC	WC	WC	WC	RO	WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-10 DMA Interrupt Clear Register 0 Description

Field	Name	Description
31:28	Reserved	Reserved
27	CH3FULLTRANSFERINTCLR	Full transfer interrupt flag clear

Field	Name	Description
26	CH3HALFTRANSFERINTCLR	Half transfer interrupt flag clear
25	CH3TRANSFERERRINTCLR	Transfer error interrupt flag clear
24	CH3DIRECTMODEERRINTCLR	Direct mode error interrupt flag clear
23	Reserved	Reserved
22	CH3FIFOERRINTCLR	FIFO error interrupt flag clear
21	CH2FULLTRANSFERINTCLR	Full tansfer interrupt flag clear
20	CH2HALFTRANSFERINTCLR	Half transfer interrupt flag clear
19	CH2TRANSFERERRINTCLR	Transfer error interrupt flag clear
18	CH2DIRECTMODEERRINTCLR	Transfer error interrupt flag clear
17	Reserved	Reserved
16	CH2FIFOERRINTCLR	Direct mode error interrupt flag clear
15:12	Reserved	Reserved
11	CH1FULLTRANSFERINTCLR	Full transfer interrupt flag clear
10	CH1HALFTRANSFERINTCLR	Half transfer interrupt flag clear
9	CH1TRANSFERERRINTCLR	Transfer error interrupt flag clear
8	CH1DIRECTMODEERRINTCLR	Direct mode error interrupt flag clear
7	Reserved	Reserved
6	CH1FIFOERRINTCLR	FIFO error interrupt flag clear
5	CH0FULLTRANSFERINTCLR	Full transfer interrupt flag clear
4	CH0HALFTRANSFERINTCLR	Half transfer interrupt flag clear
3	CH0TRANSFERERRINTCLR	Transfer error interrupt flag clear

Field	Name	Description
2	CH0DIRECTMODEERRINTCLR	Direct mode error interrupt flag clear
1	Reserved	Reserved
0	CH0FIFOERRINTCLR	FIFO error interrupt flag clear

6.5.2.4 DMA_INTCLR1

0x000c			DMA interrupt clear register 1											DMA_INTCLR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				CH7FULLTRANSFERINTCLR	CH7HALFTRANSFERINTCLR	CH7TRANSFERERRINTCLR	CH7DIRECTMODEERRINTCLR	Reserved	CH7FIFOERRINTCLR	CH6FULLTRANSFERINTCLR	CH6HALFTRANSFERINTCLR	CH6TRANSFERERRINTCLR	CH6DIRECTMODEERRINTCLR	Reserved	CH6FIFOERRINTCLR
Type	RO				WC	WC	WC	WC	RO	WC	WC	WC	WC	WC	RO	WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				CH5FULLTRANSFERINTCLR	CH5HALFTRANSFERINTCLR	CH5TRANSFERERRINTCLR	CH5DIRECTMODEERRINTCLR	Reserved	CH5FIFOERRINTCLR	CH4FULLTRANSFERINTCLR	CH4HALFTRANSFERINTCLR	CH4TRANSFERERRINTCLR	CH4DIRECTMODEERRINTCLR	Reserved	CH4FIFOERRINTCLR
Type	RO				WC	WC	WC	WC	RO	WC	WC	WC	WC	WC	RO	WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-11 DMA Interrupt Clear Register 1 Description

Field	Name	Description
31:28	Reserved	Reserved
27	CH7FULLTRANSFERINTCLR	Full transfer interrupt flag clear

Field	Name	Description
26	CH7HALFTRANSFERINTCLR	Half transfer interrupt flag clear
25	CH7TRANSFERERRINTCLR	Transfer error interrupt flag clear
24	CH7DIRECTMODEERRINTCLR	Direct mode error interrupt flag clear
23	Reserved	Reserved
22	CH7FIFOERRINTCLR	FIFO error interrupt flag clear
21	CH6FULLTRANSFERINTCLR	Full transfer interrupt flag clear
20	CH6HALFTRANSFERINTCLR	Half transfer interrupt flag clear
19	CH6TRANSFERERRINTCLR	Transfer error interrupt flag clear
18	CH6DIRECTMODEERRINTCLR	Direct mode error interrupt flag clear
17	Reserved	Reserved
16	CH6FIFOERRINTCLR	FIFO error interrupt flag clear
15:12	Reserved	Reserved
11	CH5FULLTRANSFERINTCLR	Full transfer interrupt flag clear
10	CH5HALFTRANSFERINTCLR	Half transfer interrupt flag clear
9	CH5TRANSFERERRINTCLR	Transfer error interrupt flag clear
8	CH5DIRECTMODEERRINTCLR	Direct mode error interrupt flag clear
7	Reserved	Reserved
6	CH5FIFOERRINTCLR	FIFO error interrupt flag clear
5	CH4FULLTRANSFERINTCLR	Full transfer interrupt flag clear
4	CH4HALFTRANSFERINTCLR	Half transfer interrupt flag clear
3	CH4TRANSFERERRINTCLR	Transfer error interrupt flag clear

Field	Name	Description
2	CH4DIRECTMODEERRINTCLR	Direct mode error interrupt flag clear
1:1	Reserved	Reserved
0	CH4FIFOERRINTCLR	FIFO error interrupt flag clear

6.5.2.5 DMA_CH_CFG

0x0000	DMA channel configuration register													DMA_CH_CFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				PERISRCSEL			MBURST		PBURST		Reserved	CURRENTMEM	MEMSWITCHMODE	PRIORITYLEVEL	
Type	RO				RW			RW		RW		RO	RW	RW	RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PERIINCOFFSETSIZE	MEMDATASIZE		PERIDATASIZE		MEMINCEN	PERIINC EN	CIRCULARMODE	TRANSFERDIRECTION		TRANSFERCONTROLLERSEL	FULLTRANSFERINTEN	HALFTRANSFERINTEN	TRANSFERERRINTEN	DIRECTMODEERRINTEN	EN
Type	RW	RW		RW		RW	RW	RW	RW		RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-12 DMA Channel Configuration Register Description

Field	Name	Description
31:28	Reserved	Reserved
27:25	PERISRCSEL	Channel selection These bits are set and cleared by software. 000: Request source 0 selected

Field	Name	Description
		001: Request source 1 selected 010: Request source 2 selected 011: Request source 3 selected 100: Request source 4 selected 101: Request source 5 selected 110: Request source 6 selected 111: Request source 7 selected These bits are protected and can be written only if EN is 0.
24:23	MBURST	MEM burst transfer configuration These bits are set and cleared by software. 00: Single transfer 01: INCR4 10: INCR8 11: INCR16 This field is write-protected and can be written only when EN is 0; these bits will be cleared by hardware in direct mode.
22:21	PBURST	In peripheral burst transfer configuration, these bits are set and cleared by software. 00: Single Transfer 01: Transfer with Burst Len 4 10:Transfer with Burst Len 8 11:Transfer with Burst Len 16 This field is write-protected and can be written only when EN is 0; these bits will be cleared by hardware in direct mode. Peripheral burst mode can also be used with a fixed peripheral address.
20	Reserved	Reserved

Field	Name	Description
19	CURRENTMEM	This field is only valid in MEMSWITCHMODE. 0: MEM0 is the current target memory 1: MEM1 is the current target memory This field is write-protected and can be written only when EN is 0; this field is read-only when EN is 1 to indicate which MEM is the current target memory.
18	MEMSWITCHMODE	0: Current target memory will not be switched at the end of transfer. 1: Current target memory will be switched at the end of transfer. This field is write-protected and can be written only when EN is 0.
17:16	PRIORITYLEVEL	00: Low priority 01: Medium priority 10: High priority 11: Super high priority This field is write-protected and can be written only when EN is 0.
15	PERIINCOFFSETSIZE	0: Peripheral address increment is related to PERIDATASIZE. 1: Peripheral address increment is fixed to 4. This field has no effect when ch_peri_inc_en bit is 0; this field is write-protected and can be written only when EN is 0; this field will be cleared to 0 when EN is set to 1 in case of direct mode, or PBURST is not 0.
14:13	MEMDATASIZE	These bits are set and cleared by software. 00: Byte, 8 bits 01: Half word, 16 bits 10: Word, 32 bits

Field	Name	Description
		11: Reserved This field is write-protected and can be written only when EN is 0.
12:11	PERIDATASIZE	These bits are set and cleared by software. 00: Byte, 8 bits 01: Half word, 16 bits 10: Word, 32 bits 11: Reserved This field is write-protected and can be written only when EN is 0.
10	MEMINCEN	These bits are set and cleared by software. 0: Memory address fixed 1: The memory address is incremented at the end of each transfer This field is write-protected and can be written only when EN is 0.
9	PERIINCEN	These bits are set and cleared by software. 0: Peripheral address fixed 1: The peripheral address is incremented at the end of each transfer This field is write-protected and can be written only when EN is 0.
8	CIRCULARMODE	These bits can be set and cleared by software and can also be cleared by hardware. 0: Disable cycle mode 1: Enable cycle mode; this bit will be cleared by hardware when EN is set to 1 in case of TRANSFERCONTROLLERSEL is 1.

Field	Name	Description
		This bit will be set to 1 by hardware when memory switch mode is enabled.
7:6	TRANSFERDIRECTION	These bits are set and cleared by software. 00: Peripheral to Memory 01: Memory to Peripheral 10: Memory to Memory 11: Reserved This field is write-protected and can be written only when EN is 0.
5	TRANSFERCONTROLLERSEL	0: DMA is the transfer controller 1: Peripheral is the transfer controller
4	FULLTRANSFERINTEN	0: Disable transfer complete interrupt 1: Enable transfer complete interrupt
3	HALFTRANSFERINTEN	0: Disable half of transfer complete interrupt 1: Enable half of transfer complete interrupt
2	TRANSFERERRINTEN	0: Disable transfer error interrupt 1: Enable transfer error interrupt
1	DIRECTMODEERRINTEN	0: Disable direct mode error interrupt 1: Enable direct mode error interrupt
0	EN	These bits are set and cleared by software. 0: Disable channel 1: Enable channel This bit can be cleared by hardware in the following cases: • At the end of transmission. • When AHB bus error occurs.

Field	Name	Description
		• Memory FIFO threshold is not compatible with memory burst configure.

6.5.2.6 DMA_CH_DATANUM

0x0004	DMA channel data number register													DMA_CH_DATANUM		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DATANUM															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-13 DMA Channel Data Number Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	DATANUM	The number of data to be transferred (0~65535) This field is write-protected and can be written only when EN is 0. This field is read-only when EN is 1 to indicate how much data remains. DATANUM will decrease by 1 for each data transmitted from/to the peripheral. This register will remain 0 after the transmission is completed in normal mode and will be automatically reloaded to the initial value in the cycle mode. If the value of this register is 0, no transaction can be completed even if the data flow is enabled.

6.5.2.7 DMA_CH_PERIADDR

0x0008			Peripheral base address register											DMA_CH_PERIADDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PERIADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PERIADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-14 Peripheral Base Address Register Description

Field	Name	Description
31:0	PERIADDR	The base address of the peripheral data register for reading/writing data. This field is write-protected and can be written only when EN is 0.

6.5.2.8 DMA_CH_MEM0ADDR

0x000c			MEM0 base address register											DMA_CH_MEM0ADDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	MEM0ADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MEM0ADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-15 MEM0 Base Address Register Description

Field	Name	Description
31:0	MEM0ADDR	The base address of the MEM0, for reading/writing data This field is write-protected and can be written in the following cases: • EN is 0 • EN is 1, and MEMSWITCHMODE is enabled, and CURRENTMEM is 1.

6.5.2.9 DMA_CH_MEM1ADDR

0x0010			MEM1 base address register											DMA_CH_MEM1ADDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	MEM1ADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MEM1ADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-16 MEM1 Base Address Register Description

Field	Name	Description
31:0	MEM1ADDR	The base address of the MEM1, for reading/writing data This field is available only when MEMSWITCHMODE is enabled. This field is write-protected and can be written in the following cases: • EN is 0 • EN is 1, and MEMSWITCHMODE is enabled, and CURRENTMEM is 0.

6.5.2.10 DMA_CH_FIFOCTRL

0x0014	FIFO control register													DMA_CH_FIFOCTRL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								FIFOERRINTEN	Reserved	FIFODEPTHSTS			DIRECTMODEDISABLE	FIFOTH	
Type	RO								RW	RO	RO			RW	RW	
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 6-17 FIFO Control Register Description

Field	Name	Description
31:8	Reserved	Reserved
7	FIFOERRINTEN	These bits are set and cleared by software. 0: Disable FIFO Error Interrupt 1: Enable FIFO Error Interrupt
6	Reserved	Reserved
5:3	FIFODEPTHSTS	These bits are set and cleared by hardware. 000: FIFO is not empty and the data is less than 1 word. 001: FIFO is not empty and the data is less than 2 words and more than 1 word. 010: FIFO is not empty and the data is less than 3 words and more than 2 words.

Field	Name	Description
		011: FIFO is not empty and the data is less than 4 words and more than 3 words. 100: FIFO is empty. 101: FIFO is full. 110~111: Reserved This field indicates the number of data in FIFO during transmission. This field is not used when FIFO mode is disabled.
2	DIRECTMODEDISABLE	These bits are set and cleared by software and can be set by hardware if memory to memory mode is enabled. 0: Enable direct mode 1: Disable direct mode This field is write-protected and can be written only when EN is 0.
1:0	FIFOTH	00: 1 word 01: 2 words 10: 3 words 11: 4 words This field is not used in direct mode. This field is write-protected and can be written only when EN is 0.

Nested Vectored Interrupt Controller (NVIC)

This chapter describes the details of Nested Vectored Interrupt Controller (NVIC).

Topics: **Page**

7.1 Introduction

The Nested Vector Interrupt Controller (NVIC) is a hardware component found in microcontrollers, which is responsible for managing and controlling interrupts. It enables the microcontroller to handle multiple interrupts with varying priorities efficiently.

The NVIC operates by assigning a priority level to each interrupt source. In the event of multiple interrupts occurring simultaneously, the NVIC identifies the highest priority interrupt and executes its corresponding Interrupt Service Routine (ISR) first. This ensures that critical interrupts are promptly addressed.

7.2 Features

- 116 maskable interrupt channels (excluding the sixteen Cortex ARM Star with FPU interrupt lines)
- 16 programmable priority levels (using 4 bits of interrupt priority)
- Ability to handle the low-latency exception and interrupt
- Power management control
- Support for an external Non-Maskable Interrupt (NMI)
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, allowing for low-latency interrupt processing and efficient handling of late arriving interrupts.

7.3 SysTick Calibration Value Register

The SysTick calibration value is set to 0x3E8, which gives a reference time base of 1ms with the SysTick clock set to 125 kHz.

7.4 Interrupt and Exception Vectors

Refer to the device datasheet for more details on the interrupt and exception vector table.

Cyclic Redundancy Check (CRC)

This chapter describes the details of the Cyclic Redundancy Check (CRC).

Topics: **Page**

8.1 Introduction

The Cyclic Redundancy Check (CRC) module is used to generate a signature from 8-bit, 16-bit or 32-bit data word and a generator polynomial. It is used to verify data transmission or storage integrity.

8.2 Features

- Uses CRC-32 (Ethernet):
  - CRC code: 0x4C11DB7
  - Polynomial: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-bit, 16-bit, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (suitable for temporary storage)
- Reversibility option on input/output data
- XOR option on output data

8.3 Functional Description

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and hold the result of the previous CRC calculation (read access).

Each write operation to the data register combines the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte, depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word, and right-aligned byte. For other registers, only 32-bit access is allowed.

The duration of the computation depends on the data width:

- 4 AHB clock cycles for 32-bit
- 2 AHB clock cycles for 16-bit
- 1 AHB clock cycle for 8-bit

An input buffer allows a second data to be written immediately without waiting for any wait states due to the previous CRC calculation.

The data size can be configured by the DATASIZEMODE bit in the CRC_CR register.

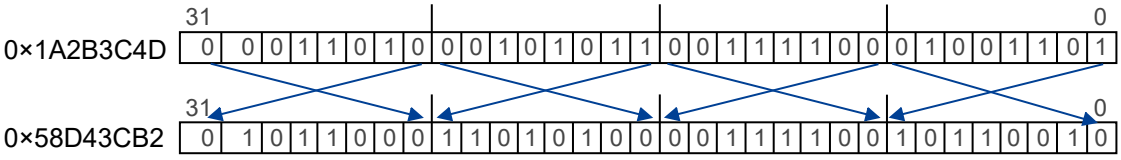
When DATASIZEMODE = 1, the data size depends on AHB access width and can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

When DATASIZEMODE = 0, the data size is defined by DATASIZE[1:0] bits in the CRC_CR register regardless of AHB access width. For example, if DATASIZE is set to 00 or 01, the data size is 32 bits. If DATASIZE is set to 10, the data size is 16 bits. If DATASIZE is set to 11, the data size is 8 bits.

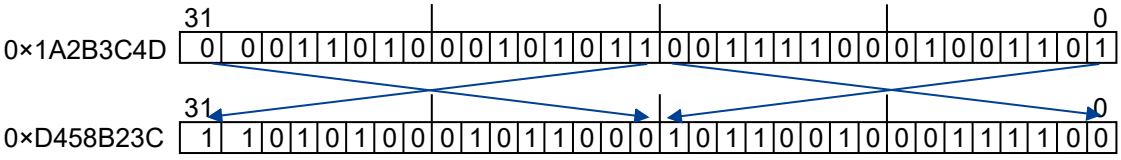
The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REVIN[1:0] bits in the CRC_CR register.

For example, the input data 0x1A2B3C4D is used for CRC calculation as:

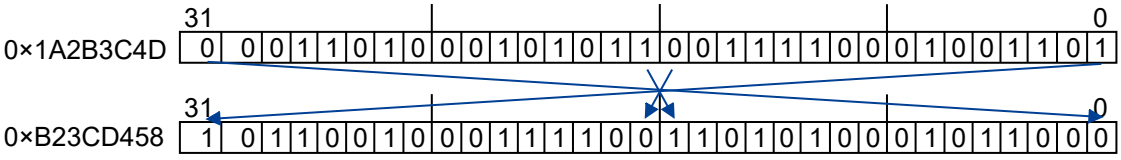
- 0x58D43CB2 with bit-reversal done by byte



- 0xD458B23C with bit-reversal done by half-word



- 0xB23CD458 with bit-reversal done on the full word



The output data can also be reversed by setting the REVOUT bit in the CRC_CR register.

The operation is done at bit level: for example, output data 0x11223344 is converted into 0x22CC4488.

The output data can be XORed by setting the XOROUT bit in the CRC_CR register.

For example, output data 0x11223344 is converted into 0xEEDDCCBB.

The CRC calculator can be initialized to a programmable value using the RST control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RST bit in the CRC_CR register.

8.3.1 Block Diagram

Figure 8-1 shows the block diagram of CRC.

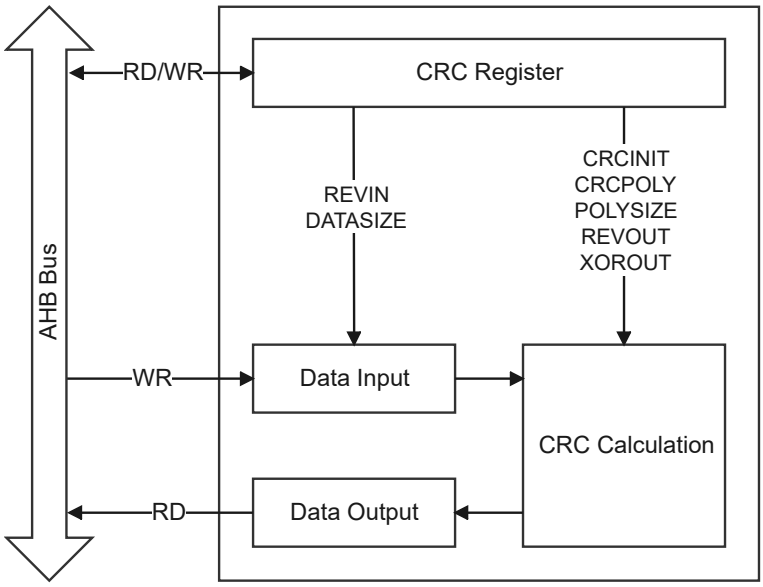


Figure 8-1 CRC Block Diagram

8.3.2 Internal Signals

Table 8-1 lists CRC internal input/output signals.

Table 8-1 CRC Internal Input/Output Signals

Signal Name	Signal Type	Description
CRC_HCLK	Digital input	AHB clock

8.3.3 Polynomial Programmability

The polynomial coefficients are fully programmable through the CRC_POLY register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[31:0] bits in the CRC_CR register. Even polynomials are not supported.

For example, set 0xA3 to CRC_POLY to use CRC-8 polynomial (X^8+ X^7+ X^5+ X +1).

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. Therefore, if a CRC calculation is ongoing, the application should either reset it or perform a CRC_DR read before changing the polynomial. The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

8.4 Registers

8.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	CRC_DR	CRC data register
0x0004	CRC_IDR	CRC independent data register
0x0008	CRC_CR	CRC control register
0x000c	CRC_INIT	CRC initial value register
0x0010	CRC_POLY	CRC polynomial register

8.4.2 Register Field Details

8.4.2.1 CRC_DR

0x0000			CRC data register											CRC_DR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DR															
Type	WR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR															
Type	WR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-2 CRC Data Register Description

Field	Name	Description
31:0	DR	CRC data register Written to this register will generate CRC data. Read from this register will get CRC results.

8.4.2.2 CRC_IDR

0x0004			CRC independent data register											CRC_IDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								IDR							
Type	RO								WR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-3 CRC Independent Data Register Description

Field	Name	Description
31:8	Reserved	Reserved
7:0	IDR	These bits can be used as a temporary storage location for 1 byte. This register is not affected by CRC resets generated by the RST bit in the CRC_CR register.

8.4.2.3 CRC_CR

0x0008			CRC control register											CRC_CR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved													XOROUT	Reserved	
Type	RO													WR	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	DATASIZE MODE	DATASIZE		Reserved				REVOUT	REVIN		POLYSIZE		Reserved		RST
Type	RO	WR	WR		RO				WR	WR		WR		RO		WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-4 CRC Control Register Description

Field	Name	Description
31:19	Reserved	Reserved
18	XOROUT	0: CRC result output without XOR 1: CRC result output with XOR XOR: All the bits in the result will be XOR-ed with 1.
17:15	Reserved	Reserved
14	DATASIZEMODE	CRC data size mode 0: Defined by bits[13:12] CR-DATASIZE 1: AHB bus HSIZE
13:12	DATASIZE	CRC data size mode 00/01: 32 bits 10: 16 bits 11: 8 bits

Field	Name	Description
11:8	Reserved	Reserved
7	REVOUT	0: CRC result output without reversed 1: CRC result output with reversed Reverse: After calculation, before XOR proc, the whole result will be swapped by bits.
6:5	REVIN	CRC data input with reversed 00: No swapped with input data 01: Swapped the input data with byte 10: Swapped the input data with half word 11: Swapped the input data with word
4:3	POLYSIZE	CRC polynomial size 00: CRC-32 01: CRC-16 10: CRC-8 11: CRC-7
2:1	Reserved	Reserved
0	RST	Used to initialize the CRC with POLYINIT by writing 1 to this bit, and it will be auto-clear. When CRC works, the user shall not write it again.

8.4.2.4 CRC_INIT

0x000c			CRC initial value register											CRC_INIT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	POLYINIT															
Type	WR															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	POLYINIT															
Type	WR															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 8-5 CRC Initial Value Register Description

Field	Name	Description
31:0	POLYINIT	Programmable initial CRC value

8.4.2.5 CRC_POLY

0x0010			CRC polynomial register											CRC_POLY		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	POLY															
Type	WR															
Reset	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	POLY															
Type	WR															
Reset	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Table 8-6 CRC Polynomial Register Description

Field	Name	Description
31:0	POLY	This register is used to write the coefficients of the polynomial to be used for CRC calculation. If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

Extended Interrupts and Events Controller (EXTI)

This chapter describes the details of the Extended Interrupts and Events Controller (EXTI).

Topics: **Page**

9.1 Introduction

The Extended Interrupts and Events Controller (EXTI) manages both external and internal asynchronous events and interrupts, and generates the event request to the CPU or interrupt controller and wake-up requests to the power controller.

9.2 Features

- Generation of up to 32 event/interrupt requests:
  - 23 configurable lines
  - 9 direct lines

  NOTE: Direct lines are not available in this product series.
- Independent masking on each event/interrupt line
- Configurable rising or falling edge (configurable lines only)
- Dedicated status bit (configurable lines only)
- Emulation of event/interrupt requests (configurable lines only)

The EXTI allows the management of up to 32 event lines that can wake up from the stop mode. These lines can be either configurable or direct.

For configurable lines:

- The active edge can be independently chosen.
- A status flag indicates the source of the interrupt.
- Configurable lines are used by I/Os external interrupts and a few peripherals.
- Each line can be masked independently for interrupt or event generation.

Additionally, this controller allows to emulate events or interrupts by software, multiplexed with the corresponding hardware event line, by writing to a dedicated register.

9.3 Functional Description

For the configurable interrupt lines, the interrupt line should be configured and enabled to generate an interrupt. 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 interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is cleared by writing a ‘1’ in the pending register.

For the direct interrupt lines, the interrupt is enabled by default in the interrupt mask register and there is no corresponding pending bit in the pending register.

To generate an 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.

For the configurable lines, an interrupt/event request can also be generated by software by writing a ‘1’ in the software interrupt/event register.

NOTE: The interrupts or events associated to the direct lines are triggered only when the system is in stop mode. If the system is still running, no interrupt/event is generated by the EXTI.

9.3.1 Block Diagram

Figure 9-1 shows the block diagram of Extended Interrupts and Events Controller (EXTI).

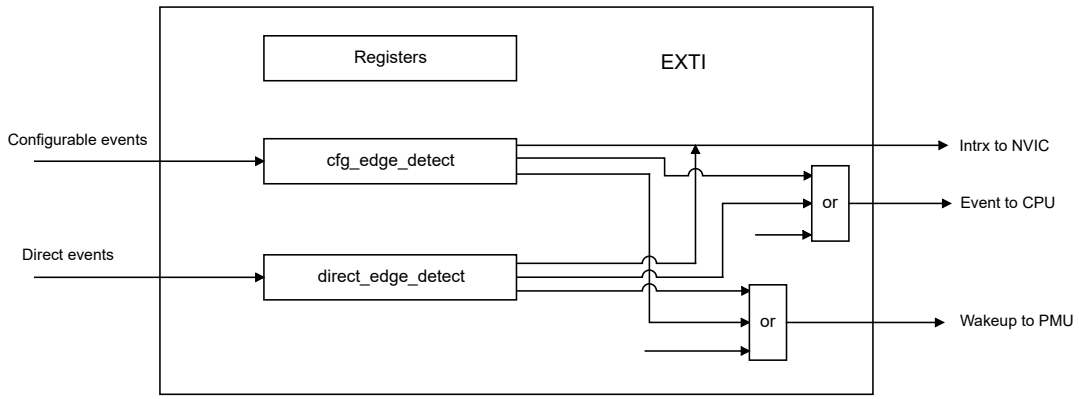


Figure 9-1 EXTI Block Diagram

9.3.2 Wakeup Event Management

Both external and internal events can be handled to wake up the core (WFE). The wakeup event can be triggered in two ways:

- By enabling an interrupt in the peripheral control register, but not in the NVIC, and enabling the SEVONPEND bit in the system control register. Upon resuming from WFE, both the EXTI peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared.
- Alternatively, by setting an EXTI line in event mode. In this case, upon resuming from WFE, there is no need to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit since the pending bit corresponding to the event line remains unset.

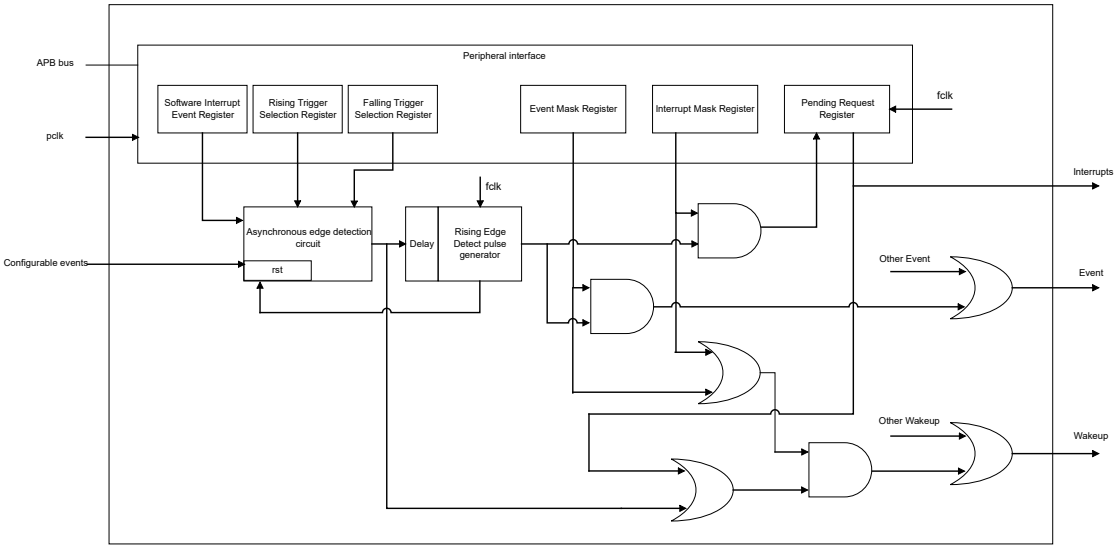


Figure 9-2 Configurable Interrupt/Event Block Diagram

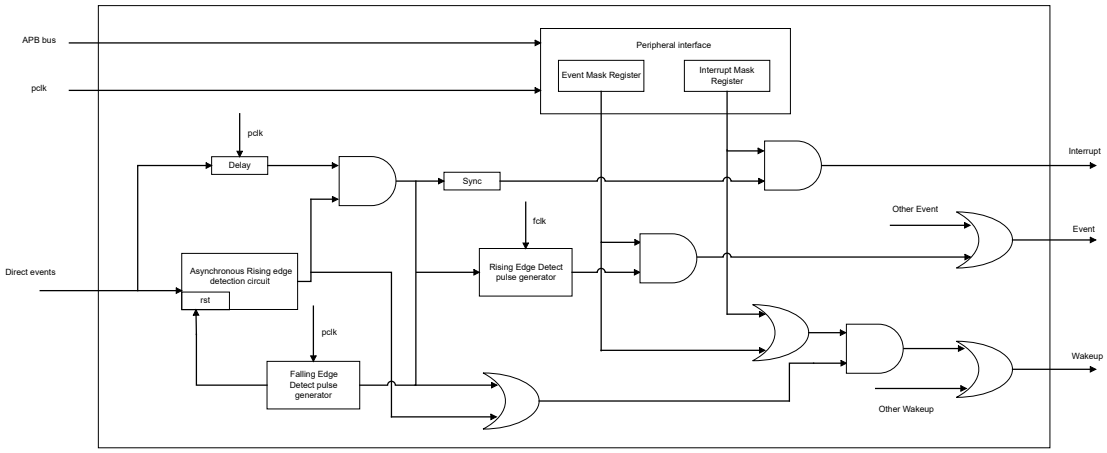


Figure 9-3 Direct Interrupt/Event Block Diagram

9.3.3 Peripherals Asynchronous Interrupts

Some peripherals can generate events when the system is in run mode and also when the system is in stop mode, allowing it to wake up the system from stop mode.

To accomplish this, the peripheral generates both a synchronized (to the system clock, such as the APB clock) and an asynchronous version of the event. The asynchronous event is then connected to an EXTI direct line.

NOTE: Few peripherals with wakeup from Stop capability are connected to an EXTI configurable line. In this case, the EXTI configuration is necessary to allow the wakeup from stop mode.

9.3.4 Hardware Interrupt Selection

To configure a line as an interrupt source, follow the steps below:

1. Configure the corresponding mask bit in the EXTI_IMR register.
2. Configure the trigger selection bits of the Interrupt line (EXTI_RTSR and EXTI_FTSR).
3. Configure the enable and mask bits that control the NVIC IRQ channel mapped to the EXTI so that an interrupt coming from one of the EXTI lines can be correctly acknowledged.

NOTE: The direct lines do not require any EXTI configuration.

9.3.5 Hardware Event Selection

To configure a line as an event source, follow the steps below:

1. Configure the corresponding mask bit in the EXTI_EMR register.
2. Configure the trigger selection bits of the event line (EXTI_RTSR and EXTI_FTSR).

9.3.6 Software Interrupt/Event Selection

Any of the configurable lines can be configured as a software interrupt/event line. The procedure to generate a software interrupt is as follows:

1. Configure the corresponding mask bit (EXTI_IMR, EXTI_EMR).
2. Set the required bit of the software interrupt register (EXTI_SWIER).

9.4 Interrupt/Event Line Mapping

There are 32 accessible interrupt/event lines. The GPIOs are connected to 16 configurable interrupt/event lines (refer to Figure 9-4).

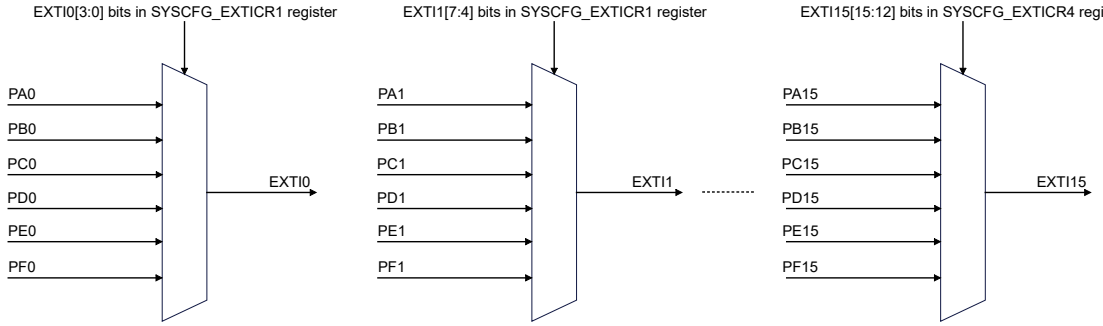


Figure 9-4 External Interrupt/Event GPIO Mapping

The EXTI lines are connected as shown in Table 9-1.

Table 9-1 EXTI Lines Connections

EXTI Line	Line Source	Line Type
0	EXTI0	Configurable
1	EXTI1	Configurable
2	EXTI2	Configurable
3	EXTI3	Configurable
4	EXTI4	Configurable
5	EXTI5	Configurable
6	EXTI6	Configurable
7	EXTI7	Configurable
8	EXTI8	Configurable
9	EXTI9	Configurable
10	EXTI10	Configurable
11	EXTI11	Configurable
12	EXTI12	Configurable
13	EXTI13	Configurable
14	EXTI14	Configurable
15	EXTI15	Configurable
16	SVS	Configurable
17	Reserved	Configurable
18	Timestamp or CSS_ELS	Configurable
19	RTC wakeup timer	Configurable
20	CMP0 output	Configurable

EXTI Line	Line Source	Line Type
21	CMP1 output	Configurable
31	TPSensor	Configurable

9.5 Registers

9.5.1 Register Address Map

Offset	Register Name	Register Description
0x0000	EXTI_IMR1	Interrupt mask register 1
0x0004	EXTI_EMR1	Event mask register 1
0x0008	EXTI_RTSR1	Rising trigger selection register 1
0x000c	EXTI_FTSR1	Falling trigger selection register 1
0x0010	EXTI_SWIER1	Software interrupt event register 1
0x0014	EXTI_PR1	Pending register 1

9.5.2 Register Field Details

9.5.2.1 EXTI_IMR1

0x0000			Interrupt mask register 1											EXTI_IMR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-2 Interrupt Mask Register 1 Description

Field	Name	Description
31	IM31	Interrupt Mask on line 31 0: Interrupt request from Line 31 is masked 1: Interrupt request from Line 31 is not masked
30	IM30	Interrupt Mask on line 30 0: Interrupt request from Line 30 is masked 1: Interrupt request from Line 30 is not masked
29	IM29	Interrupt Mask on line 29 0: Interrupt request from Line 29 is masked 1: Interrupt request from Line 29 is not masked
28	IM28	Interrupt Mask on line 28 0: Interrupt request from Line 28 is masked

Field	Name	Description
		1: Interrupt request from Line 28 is not masked
27	IM27	Interrupt Mask on line 27 0: Interrupt request from Line 27 is masked 1: Interrupt request from Line 27 is not masked
26	IM26	Interrupt Mask on line 26 0: Interrupt request from Line 26 is masked 1: Interrupt request from Line 26 is not masked
25	IM25	Interrupt Mask on line 25 0: Interrupt request from Line 25 is masked 1: Interrupt request from Line 25 is not masked
24	IM24	Interrupt Mask on line 24 0: Interrupt request from Line 24 is masked 1: Interrupt request from Line 24 is not masked
23	IM23	Interrupt Mask on line 23 0: Interrupt request from Line 23 is masked 1: Interrupt request from Line 23 is not masked
22	IM22	Interrupt Mask on line 22 0: Interrupt request from Line 22 is masked 1: Interrupt request from Line 22 is not masked
21	IM21	Interrupt Mask on line 21 0: Interrupt request from Line 21 is masked 1: Interrupt request from Line 21 is not masked
20	IM20	Interrupt Mask on line 20 0: Interrupt request from Line 20 is masked 1: Interrupt request from Line 20 is not masked

Field	Name	Description
19	IM19	Interrupt Mask on line 19 0: Interrupt request from Line 19 is masked 1: Interrupt request from Line 19 is not masked
18	IM18	Interrupt Mask on line 18 0: Interrupt request from Line 18 is masked 1: Interrupt request from Line 18 is not masked
17	IM17	Interrupt Mask on line 17 0: Interrupt request from Line 17 is masked 1: Interrupt request from Line 17 is not masked
16	IM16	Interrupt Mask on line 16 0: Interrupt request from Line 16 is masked 1: Interrupt request from Line 16 is not masked
15	IM15	Interrupt Mask on line 15 0: Interrupt request from Line 15 is masked 1: Interrupt request from Line 15 is not masked
14	IM14	Interrupt Mask on line 14 0: Interrupt request from Line 14 is masked 1: Interrupt request from Line 14 is not masked
13	IM13	Interrupt Mask on line 13 0: Interrupt request from Line 13 is masked 1: Interrupt request from Line 13 is not masked
12	IM12	Interrupt Mask on line 12 0: Interrupt request from Line 12 is masked 1: Interrupt request from Line 12 is not masked
11	IM11	Interrupt Mask on line 11

Field	Name	Description
		0: Interrupt request from Line 11 is masked 1: Interrupt request from Line 11 is not masked
10	IM10	Interrupt Mask on line 10 0: Interrupt request from Line 10 is masked 1: Interrupt request from Line 10 is not masked
9	IM9	Interrupt Mask on line 9 0: Interrupt request from Line 9 is masked 1: Interrupt request from Line 9 is not masked
8	IM8	Interrupt Mask on line 8 0: Interrupt request from Line 8 is masked 1: Interrupt request from Line 8 is not masked
7	IM7	Interrupt Mask on line 7 0: Interrupt request from Line 7 is masked 1: Interrupt request from Line 7 is not masked
6	IM6	Interrupt Mask on line 6 0: Interrupt request from Line 6 is masked 1: Interrupt request from Line 6 is not masked
5	IM5	Interrupt Mask on line 5 0: Interrupt request from Line 5 is masked 1: Interrupt request from Line 5 is not masked
4	IM4	Interrupt Mask on line 4 0: Interrupt request from Line 4 is masked 1: Interrupt request from Line 4 is not masked
3	IM3	Interrupt Mask on line 3 0: Interrupt request from Line 3 is masked

Field	Name	Description
		1: Interrupt request from Line 3 is not masked
2	IM2	Interrupt Mask on line 2 0: Interrupt request from Line 2 is masked 1: Interrupt request from Line 2 is not masked
1	IM1	Interrupt Mask on line 1 0: Interrupt request from Line 1 is masked 1: Interrupt request from Line 1 is not masked
0	IM0	Interrupt Mask on line 0 0: Interrupt request from Line 0 is masked 1: Interrupt request from Line 0 is not masked

9.5.2.2 EXTI_EMR1

0x0004			Event mask register 1											EXTI_EMR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-3 Event Mask Register 1Description

Field	Name	Description
31	EM31	Event Mask on line 31 0: Event request from Line 31 is masked 1: Event request from Line 31 is not masked
30	EM30	Event Mask on line 30 0: Event request from Line 30 is masked 1: Event request from Line 30 is not masked
29	EM29	Event Mask on line 29 0: Event request from Line 29 is masked 1: Event request from Line 29 is not masked
28	EM28	Event Mask on line 28 0: Event request from Line 28 is masked 1: Event request from Line 28 is not masked

Field	Name	Description
27	EM27	Event Mask on line 27 0: Event request from Line 27 is masked 1: Event request from Line 27 is not masked
26	EM26	Event Mask on line 26 0: Event request from Line 26 is masked 1: Event request from Line 26 is not masked
25	EM25	Event Mask on line 25 0: Event request from Line 25 is masked 1: Event request from Line 25 is not masked
24	EM24	Event Mask on line 24 0: Event request from Line 24 is masked 1: Event request from Line 24 is not masked
23	EM23	Event Mask on line 23 0: Event request from Line 23 is masked 1: Event request from Line 23 is not masked
22	EM22	Event Mask on line 22 0: Event request from Line 22 is masked 1: Event request from Line 22 is not masked
21	EM21	Event Mask on line 21 0: Event request from Line 21 is masked 1: Event request from Line 21 is not masked
20	EM20	Event Mask on line 20 0: Event request from Line 20 is masked 1: Event request from Line 20 is not masked
19	EM19	Event Mask on line 19

Field	Name	Description
		0: Event request from Line 19 is masked 1: Event request from Line 19 is not masked
18	EM18	Event Mask on line 18 0: Event request from Line 18 is masked 1: Event request from Line 18 is not masked
17	EM17	Event Mask on line 17 0: Event request from Line 17 is masked 1: Event request from Line 17 is not masked
16	EM16	Event Mask on line 16 0: Event request from Line 16 is masked 1: Event request from Line 16 is not masked
15	EM15	Event Mask on line 15 0: Event request from Line 15 is masked 1: Event request from Line 15 is not masked
14	EM14	Event Mask on line 14 0: Event request from Line 14 is masked 1: Event request from Line 14 is not masked
13	EM13	Event Mask on line 13 0: Event request from Line 13 is masked 1: Event request from Line 13 is not masked
12	EM12	Event Mask on line 12 0: Event request from Line 12 is masked 1: Event request from Line 12 is not masked
11	EM11	Event Mask on line 11 0: Event request from Line 11 is masked

Field	Name	Description
		1: Event request from Line 11 is not masked
10	EM10	Event Mask on line 10 0: Event request from Line 10 is masked 1: Event request from Line 10 is not masked
9	EM9	Event Mask on line 9 0: Event request from Line 9 is masked 1: Event request from Line 9 is not masked
8	EM8	Event Mask on line 8 0: Event request from Line 8 is masked 1: Event request from Line 8 is not masked
7	EM7	Event Mask on line 7 0: Event request from Line 7 is masked 1: Event request from Line 7 is not masked
6	EM6	Event Mask on line 6 0: Event request from Line 6 is masked 1: Event request from Line 6 is not masked
5	EM5	Event Mask on line 5 0: Event request from Line 5 is masked 1: Event request from Line 5 is not masked
4	EM4	Event Mask on line 4 0: Event request from Line 4 is masked 1: Event request from Line 4 is not masked
3	EM3	Event Mask on line 3 0: Event request from Line 3 is masked 1: Event request from Line 3 is not masked

Field	Name	Description
2	EM2	Event Mask on line 2 0: Event request from Line 2 is masked 1: Event request from Line 2 is not masked
1	EM1	Event Mask on line 1 0: Event request from Line 1 is masked 1: Event request from Line 1 is not masked
0	EM0	Event Mask on line 0 0: Event request from Line 0 is masked 1: Event request from Line 0 is not masked

9.5.2.3 EXTI_RTSR1

0x0008			Rising trigger selection register 1											EXTI_RTSR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RT31	Reserved									RT21	RT20	RT19	RT18	RT17	RT16
Type	RW	RO									RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-4 Rising Trigger Selection Register 1 Description

Field	Name	Description
31	RT31	Rising trigger event configuration bit of line 31 0: Rising trigger disabled (for event and interrupt) for input line 31 1: Rising trigger enabled (for event and interrupt) for input line 31
30:22	Reserved	Reserved
21	RT21	Rising trigger event configuration bit of line 21 0: Rising trigger disabled (for event and interrupt) for input line 21 1: Rising trigger enabled (for event and interrupt) for input line 21
20	RT20	Rising trigger event configuration bit of line 20 0: Rising trigger disabled (for event and interrupt) for input line 20 1: Rising trigger enabled (for event and interrupt) for input line 20
19	RT19	Rising trigger event configuration bit of line 19 0: Rising trigger disabled (for event and interrupt) for input line 19

Field	Name	Description
		1: Rising trigger enabled (for event and interrupt) for input line 19
18	RT18	Rising trigger event configuration bit of line 18 0: Rising trigger disabled (for event and interrupt) for input line 18 1: Rising trigger enabled (for event and interrupt) for input line 18
17	RT17	Rising trigger event configuration bit of line 17 0: Rising trigger disabled (for event and interrupt) for input line 17 1: Rising trigger enabled (for event and interrupt) for input line 17
16	RT16	Rising trigger event configuration bit of line 16 0: Rising trigger disabled (for event and interrupt) for input line 16 1: Rising trigger enabled (for event and interrupt) for input line 16
15	RT15	Rising trigger event configuration bit of line 15 0: Rising trigger disabled (for event and interrupt) for input line 15 1: Rising trigger enabled (for event and interrupt) for input line 15
14	RT14	Rising trigger event configuration bit of line 14 0: Rising trigger disabled (for event and interrupt) for input line 14 1: Rising trigger enabled (for event and interrupt) for input line 14
13	RT13	Rising trigger event configuration bit of line 13 0: Rising trigger disabled (for event and interrupt) for input line 13 1: Rising trigger enabled (for event and interrupt) for input line 13
12	RT12	Rising trigger event configuration bit of line 12 0: Rising trigger disabled (for event and interrupt) for input line 12 1: Rising trigger enabled (for event and interrupt) for input line 12
11	RT11	Rising trigger event configuration bit of line 11 0: Rising trigger disabled (for event and interrupt) for input line 11 1: Rising trigger enabled (for event and interrupt) for input line 11

Field	Name	Description
10	RT10	Rising trigger event configuration bit of line 10 0: Rising trigger disabled (for event and interrupt) for input line 10 1: Rising trigger enabled (for event and interrupt) for input line 10
9	RT9	Rising trigger event configuration bit of line 9 0: Rising trigger disabled (for event and interrupt) for input line 9 1: Rising trigger enabled (for event and interrupt) for input line 9
8	RT8	Rising trigger event configuration bit of line 8 0: Rising trigger disabled (for event and interrupt) for input line 8 1: Rising trigger enabled (for event and interrupt) for input line 8
7	RT7	Rising trigger event configuration bit of line 7 0: Rising trigger disabled (for event and interrupt) for input line 7 1: Rising trigger enabled (for event and interrupt) for input line 7
6	RT6	Rising trigger event configuration bit of line 6 0: Rising trigger disabled (for event and interrupt) for input line 6 1: Rising trigger enabled (for event and interrupt) for input line 6
5	RT5	Rising trigger event configuration bit of line 5 0: Rising trigger disabled (for event and interrupt) for input line 5 1: Rising trigger enabled (for event and interrupt) for input line 5
4	RT4	Rising trigger event configuration bit of line 4 0: Rising trigger disabled (for event and interrupt) for input line 4 1: Rising trigger enabled (for event and interrupt) for input line 4
3	RT3	Rising trigger event configuration bit of line 3 0: Rising trigger disabled (for event and interrupt) for input line 3 1: Rising trigger enabled (for event and interrupt) for input line 3
2	RT2	Rising trigger event configuration bit of line 2

Field	Name	Description
		0: Rising trigger disabled (for event and interrupt) for input line 2 1: Rising trigger enabled (for event and interrupt) for input line 2
1	RT1	Rising trigger event configuration bit of line 1 0: Rising trigger disabled (for event and interrupt) for input line 1 1: Rising trigger enabled (for event and interrupt) for input line 1
0	RT0	Rising trigger event configuration bit of line 0 0: Rising trigger disabled (for event and interrupt) for input line 0 1: Rising trigger enabled (for event and interrupt) for input line 0

9.5.2.4 EXTI_FTSR1

0x000c			Falling trigger selection register 1									EXTI_FTSR1				
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	FT31	Reserved									FT21	FT20	FT19	FT18	FT17	FT16
Type	RW	RO									RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-5 Falling Trigger Selection Register 1 Description

Field	Name	Description
31	FT31	Falling trigger event configuration bit of line 31 0: Falling trigger disabled (for event and interrupt) for input line 31 1: Falling trigger enabled (for event and interrupt) for input line 31
30:22	Reserved	Reserved
21	FT21	Falling trigger event configuration bit of line 21 0: Falling trigger disabled (for event and interrupt) for input line 21 1: Falling trigger enabled (for event and interrupt) for input line 21
20	FT20	Falling trigger event configuration bit of line 20 0: Falling trigger disabled (for event and interrupt) for input line 20 1: Falling trigger enabled (for event and interrupt) for input line 20
19	FT19	Falling trigger event configuration bit of line 19 0: Falling trigger disabled (for event and interrupt) for input line 19

Field	Name	Description
		1: Falling trigger enabled (for event and interrupt) for input line 19
18	FT18	Falling trigger event configuration bit of line 18 0: Falling trigger disabled (for event and interrupt) for input line 18 1: Falling trigger enabled (for event and interrupt) for input line 18
17	FT17	Falling trigger event configuration bit of line 17 0: Falling trigger disabled (for event and interrupt) for input line 17 1: Falling trigger enabled (for event and interrupt) for input line 17
16	FT16	Falling trigger event configuration bit of line 16 0: Falling trigger disabled (for event and interrupt) for input line 16 1: Falling trigger enabled (for event and interrupt) for input line 16
15	FT15	Falling trigger event configuration bit of line 15 0: Falling trigger disabled (for event and interrupt) for input line 15 1: Falling trigger enabled (for event and interrupt) for input line 15
14	FT14	Falling trigger event configuration bit of line 14 0: Falling trigger disabled (for event and interrupt) for input line 14 1: Falling trigger enabled (for event and interrupt) for input line 14
13	FT13	Falling trigger event configuration bit of line 13 0: Falling trigger disabled (for event and interrupt) for input line 13 1: Falling trigger enabled (for event and interrupt) for input line 13
12	FT12	Falling trigger event configuration bit of line 12 0: Falling trigger disabled (for event and interrupt) for input line 12 1: Falling trigger enabled (for event and interrupt) for input line 12
11	FT11	Falling trigger event configuration bit of line 11 0: Falling trigger disabled (for event and interrupt) for input line 11 1: Falling trigger enabled (for event and interrupt) for input line 11

Field	Name	Description
10	FT10	Falling trigger event configuration bit of line 10 0: Falling trigger disabled (for event and interrupt) for input line 10 1: Falling trigger enabled (for event and interrupt) for input line 10
9	FT9	Falling trigger event configuration bit of line 9 0: Falling trigger disabled (for event and interrupt) for input line 9 1: Falling trigger enabled (for event and interrupt) for input line 9
8	FT8	Falling trigger event configuration bit of line 8 0: Falling trigger disabled (for event and interrupt) for input line 8 1: Falling trigger enabled (for event and interrupt) for input line 8
7	FT7	Falling trigger event configuration bit of line 7 0: Falling trigger disabled (for event and interrupt) for input line 7 1: Falling trigger enabled (for event and interrupt) for input line 7
6	FT6	Falling trigger event configuration bit of line 6 0: Falling trigger disabled (for event and interrupt) for input line 6 1: Falling trigger enabled (for event and interrupt) for input line 6
5	FT5	Falling trigger event configuration bit of line 5 0: Falling trigger disabled (for event and interrupt) for input line 5 1: Falling trigger enabled (for event and interrupt) for input line 5
4	FT4	Falling trigger event configuration bit of line 4 0: Falling trigger disabled (for event and interrupt) for input line 4 1: Falling trigger enabled (for event and interrupt) for input line 4
3	FT3	Falling trigger event configuration bit of line 3 0: Falling trigger disabled (for event and interrupt) for input line 3 1: Falling trigger enabled (for event and interrupt) for input line 3
2	FT2	Falling trigger event configuration bit of line 2

Field	Name	Description
		0: Falling trigger disabled (for event and interrupt) for input line 2 1: Falling trigger enabled (for event and interrupt) for input line 2
1	FT1	Falling trigger event configuration bit of line 1 0: Falling trigger disabled (for event and interrupt) for input line 1 1: Falling trigger enabled (for event and interrupt) for input line 1
0	FT0	Falling trigger event configuration bit of line 0 0: Falling trigger disabled (for event and interrupt) for input line 0 1: Falling trigger enabled (for event and interrupt) for input line 0

9.5.2.5 EXTI_SWIER1

0x0010			Software interrupt event register 1											EXTI_SWIER1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	SWI31	Reserved									SWI21	SWI20	SWI19	SWI18	SWI17	SWI16
Type	RW	RO									RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-6 Software Interrupt Event Register 1 Description

Field	Name	Description
31	SWI31	Software rising edge event trigger on line 31 The setting of any bit by software triggers a rising edge event on the corresponding line 31, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
30:22	Reserved	Reserved
21	SWI21	Software rising edge event trigger on line 21 The setting of any bit by software triggers a rising edge event on the corresponding line 21, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
20	SWI20	Software rising edge event trigger on line 20

Field	Name	Description
		The setting of any bit by software triggers a rising edge event on the corresponding line 20, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
19	SWI19	Software rising edge event trigger on line 19 The setting of any bit by software triggers a rising edge event on the corresponding line 19, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
18	SWI18	Software rising edge event trigger on line 18 The setting of any bit by software triggers a rising edge event on the corresponding line 18, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
17	SWI17	Software rising edge event trigger on line 17 The setting of any bit by software triggers a rising edge event on the corresponding line 17, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
16	SWI16	Software rising edge event trigger on line 16 The setting of any bit by software triggers a rising edge event on the corresponding line 16, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
15	SWI15	Software rising edge event trigger on line 15 The setting of any bit by software triggers a rising edge event on the corresponding line 15, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
14	SWI14	Software rising edge event trigger on line 14 The setting of any bit by software triggers a rising edge event on the corresponding line 14, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings.

Field	Name	Description
		The bits are automatically cleared by hardware. Reading of any bit always returns 0.
13	SWI13	Software rising edge event trigger on line 13 The setting of any bit by software triggers a rising edge event on the corresponding line 13, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
12	SWI12	Software rising edge event trigger on line 12 The setting of any bit by software triggers a rising edge event on the corresponding line 12, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
11	SWI11	Software rising edge event trigger on line 11 The setting of any bit by software triggers a rising edge event on the corresponding line 11, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
10	SWI10	Software rising edge event trigger on line 10 The setting of any bit by software triggers a rising edge event on the corresponding line 10, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
9	SWI9	Software rising edge event trigger on line 9 The setting of any bit by software triggers a rising edge event on the corresponding line 9, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
8	SWI8	Software rising edge event trigger on line 8 The setting of any bit by software triggers a rising edge event on the corresponding line 8, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
7	SWI7	Software rising edge event trigger on line 7

Field	Name	Description
		The setting of any bit by software triggers a rising edge event on the corresponding line 7, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
6	SWI6	Software rising edge event trigger on line 6 The setting of any bit by software triggers a rising edge event on the corresponding line 6, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
5	SWI5	Software rising edge event trigger on line 5 The setting of any bit by software triggers a rising edge event on the corresponding line 5, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
4	SWI4	Software rising edge event trigger on line 4 The setting of any bit by software triggers a rising edge event on the corresponding line 4, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
3	SWI3	Software rising edge event trigger on line 3 The setting of any bit by software triggers a rising edge event on the corresponding line 3, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
2	SWI2	Software rising edge event trigger on line 2 The setting of any bit by software triggers a rising edge event on the corresponding line 2, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.
1	SWI1	Software rising edge event trigger on line 1 The setting of any bit by software triggers a rising edge event on the corresponding line 1, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings.

Field	Name	Description
		The bits are automatically cleared by hardware. Reading of any bit always returns 0.
0	SWI0	Software rising edge event trigger on line 0 The setting of any bit by software triggers a rising edge event on the corresponding line 0, resulting in an interrupt, independently of EXTI_RTSR and EXTI_FTSR settings. The bits are automatically cleared by hardware. Reading of any bit always returns 0.

9.5.2.6 EXTI_PR1

0x0014			Pending register 1											EXTI_PR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PIF31	Reserved									PIF21	PIF20	PIF19	PIF18	PIF17	PIF16
Type	RC_W1	RO									RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PIF15	PIF14	PIF13	PIF12	PIF11	PIF10	PIF9	PIF8	PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
Type	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 9-7 Pending Register 1 Description

Field	Name	Description
31	PIF31	Pending interrupt flag on line 31 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
30:22	Reserved	Reserved
21	PIF21	Pending interrupt flag on line 21 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
20	PIF20	Pending interrupt flag on line 20 0: No trigger request occurred

Field	Name	Description
		1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
19	PIF19	Pending interrupt flag on line 19 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
18	PIF18	Pending interrupt flag on line 18 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
17	PIF17	Pending interrupt flag on line 17 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
16	PIF16	Pending interrupt flag on line 16 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
15	PIF15	Pending interrupt flag on line 15 0: No trigger request occurred 1: Selected trigger request occurred

Field	Name	Description
		This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
14	PIF14	Pending interrupt flag on line 14 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit
13	PIF13	Pending interrupt flag on line 13 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
12	PIF12	Pending interrupt flag on line 12 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
11	PIF11	Pending interrupt flag on line 11 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
10	PIF10	Pending interrupt flag on line 10 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line.

Field	Name	Description
		This bit is cleared by writing a '1' to the bit
9	PIF9	Pending interrupt flag on line 9 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit
8	PIF8	Pending interrupt flag on line 8 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
7	PIF7	Pending interrupt flag on line 7 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit
6	PIF6	Pending interrupt flag on line 6 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
5	PIF5	Pending interrupt flag on line 5 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.

Field	Name	Description
4	PIF4	Pending interrupt flag on line 4 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
3	PIF3	Pending interrupt flag on line 3 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
2	PIF2	Pending interrupt flag on line 2 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
1	PIF1	Pending interrupt flag on line 1 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.
0	PIF0	Pending interrupt flag on line 0 0: No trigger request occurred 1: Selected trigger request occurred This bit is set when the selected edge event arrives on the interrupt line. This bit is cleared by writing a '1' to the bit.

General-Purpose I/Os (GPIO)

This chapter describes the details of the General-Purpose I/Os (GPIO).

Topics: **Page**

10.1 Introduction

Each General-Purpose I/O (GPIO) port has seven 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_DRR, GPIOx_DFR, GPIOx_ASR and GPIOx_CSR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR) and a 32-bit set/reset register (GPIOx_BSRR) and 32-bit reset register (GPIOx_BRR).

All I/O ports have the one 32-bit lock registers (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL). In addition, all the I/O ports also have two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL). It also has a 32-bit analog switch register (GPIOx_ASR) for GPIO with an analog switch function.

10.2 Features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

10.3 Functional Description

Depending on the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the General-Purpose I/O (GPIO) ports can be individually configured by software in several modes:

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

Each I/O port bit is freely programmable; however, the I/O port registers can only be accessed as 32-bit words. The purpose of the GPIOx_BSRR and GPIOx_BRR register is to allow modify accesses to any bits of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 10-1 shows the basic structures of a standard tolerant I/O port bit, respectively. Table 10-1 gives the possible port bit configurations.

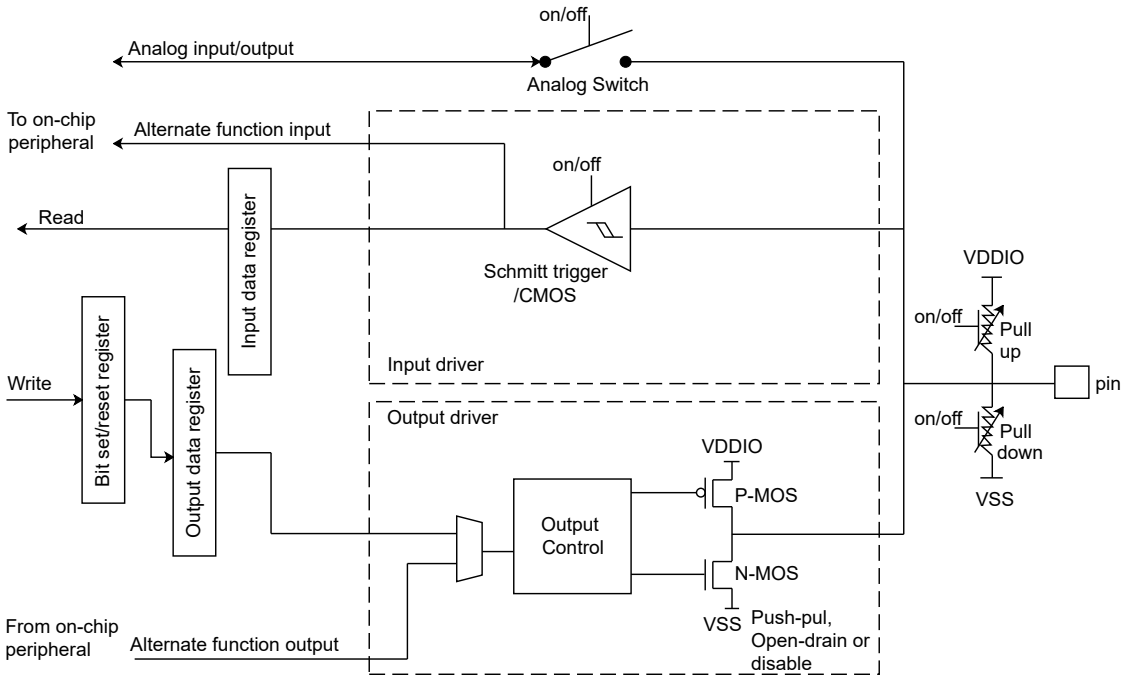


Figure 10-1 Basic Structure of an I/O Port Bit

Table 10-1 Port Bit Configuration

MODER [1:0]	OTYPER	OSPEEDR		PUPDR[1:0]		I/O Configuration	
01	0	OSPEEDRy[1:0]		0	0	GP (1)output	PP (2)
	0			0	1	GP output	PP + PU (1)
	0			1	0	GP output	PP + PD(4)
	0			1	1	Reserved	
	1			0	0	GP output	OD (5)
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved	
10	0	OSPEEDRy[1:0]		0	0	AF (6)	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	

MODER [1:0]	OTYPER	OSPEEDR		PUPDR[1:0]		I/O Configuration	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Input/output	Analog, PU
	x	x	x	1	0	Input/output	Analog, PD
	x	x	x	1	1	Reserved	

(1) GP: General-purpose
(2) PP: Push-pull
(3) PU: Pull-up
(4) PD: Pull-down
(5) OD: Open-drain
(6) AF: Alternate function

In low-power standby mode, all the state of I/O pins is kept High-Z, except the below I/O pins:

- RESET pin
- Wakeup pins

10.3.1 General-Purpose I/O (GPIO)

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

The debug pins are in AF pull-up/pull-down after reset:

- JTDI: Pull-up
- JTCK/SWCLK: Pull-down
- JTMS/SWDIO: Pull-up
- NJTRST: Pull-up
- JTDO: Floating state no pull-up/pull-down

PE11/BOOT is in input mode during the reset until at least the end of the option byte loading phase. Refer to Using PE11 as GPIO.

PE11 can be used as BOOT pin or one GPIO, which is determined by the option byte of NSW_BOOT0 bit in FMC_OPTR register, it is set from input mode to analog mode based on the condition below:

- It is loaded after the option byte if nSWBOOT0 = 1.
- It is after reset if nSWBOOT0= 0.

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB2 clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or disconnected.

10.3.2 I/O Pin Alternate Function Multiplexer and Mapping

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

Each I/O pin has a multiplexer with up to eight AF inputs (AF0 to AF7) that can be configured through the GPIOx_AFRL and GPIOx_AFRH registers:

- After reset the multiplexer selection is AF0. The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.
  - In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, follow these steps:

- Debug function: After each device reset, these pins are assigned as alternate function pins immediately usable by the debugger host.
- GPIO: Configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- Peripheral alternate function:
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_OSPEEDR, and GPIOx_PUPDR registers, respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- Additional functions:
  - For the WKUPx, ADC, DAC, OA, CMP and TPSensor, configure the desired I/O in analog mode in the GPIOx_MODER register and enable analog switch for some of analog functions in the GPIOx_ASR register, and configure the required function in the ADC, DAC, OA, CMP and TPSensor registers.

    NOTE: Analog switch is not enabled with WKUPx, DAC and OA, please refer to device datasheet for which port should be used.

  - For the additional functions like RTC and oscillators, configure the required function in the related RTC and Reset & Clock Control registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate Functions (AF0-AF7)” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

10.3.3 I/O Port Control Registers

Each GPIO port contains four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR), allowing configuration of up to 16 I/Os.

The GPIOx_MODER register is used to select the I/O mode, which can be input, output, AF, or analog. The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and IO speed rate. The GPIOx_PUPDR register is used to select the pull-up/pull-down, regardless of the I/O direction.

10.3.4 I/O Port Data Registers

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

For detailed descriptions of these registers, refer to GPIOx_IDR (x = A to F) and GPIOx_ODR (x = A to F).

10.3.5 I/O Data Bitwise Handling

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

Each bit in the GPIOx_ODR corresponds to two control bits in the GPIOx_BSRR: the bit set (BS) and bit reset (BR). When the BS[15:0] bits are written to 1, the corresponding ODR bit is set by the BS bit. When the BR[31:16] bits are written to 1, the corresponding ODR bit is reset by the BR bit.

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

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

When programming the GPIOx_ODR at the bit level, there is no need for the software to disable interrupts. It is possible to modify one or more bits in a single atomic AHB write access.

10.3.6 GPIO Locking Mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH, GPIOx_DFR, GPIOx_ASR and GPIOx_CSR.

To write the GPIOx_LCKR register, a specific write/read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence, the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH, GPIOx_DFR, GPIOx_ASR and GPIOx_CSR).

The LOCK sequence can only be performed using a word (32-bit long) access to the GPIOx_LCKR register because GPIOx_LCKR bit 16 has to be set simultaneously as the [15:0] bits. For more details, refer to the description in the GPIO port configuration lock register (GPIOx_LCKR) (x = A to F).

NOTE: The GPIO locking mechanism can only operate one time after MCU reset.

10.3.7 Alternate Function I/O

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

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

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

10.3.8 External Interrupt/Wakeup Lines

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

For more details, refer to Wakeup Event Management.

10.3.9 Input Configuration

When the I/O port is programmed as input:

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

Figure 10-2 shows the input configuration of the I/O port bit.

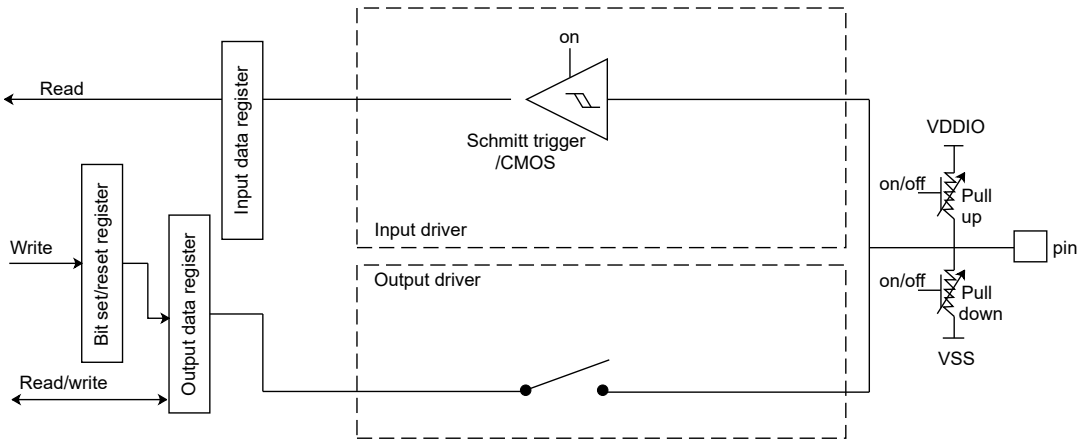


Figure 10-2 Input Configuration

10.3.10 Output Configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in High-Z (P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input or CMOS input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- In open drain mode, a read access to the input data register gets the I/O state
- In push-pull mode, a read access to the output data register gets the last written value
- For the output speed on GPIO, it can be configured with the register GPIOx_OSPEEDR
- For GPIO driving strength control, it can be configured with the register GPIOx_DRR

Figure 10-3 shows the output configuration of the I/O port bit.

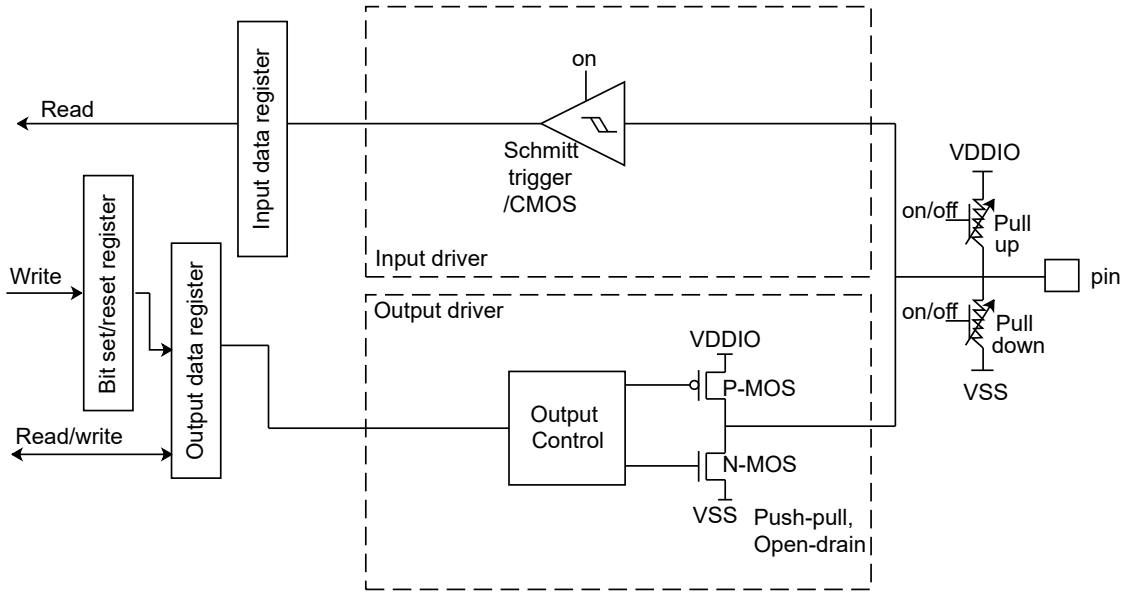


Figure 10-3 Output Configuration

10.3.11 Alternate Function Configuration

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

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

Figure 10-4 shows the alternate function configuration of the I/O port bit.

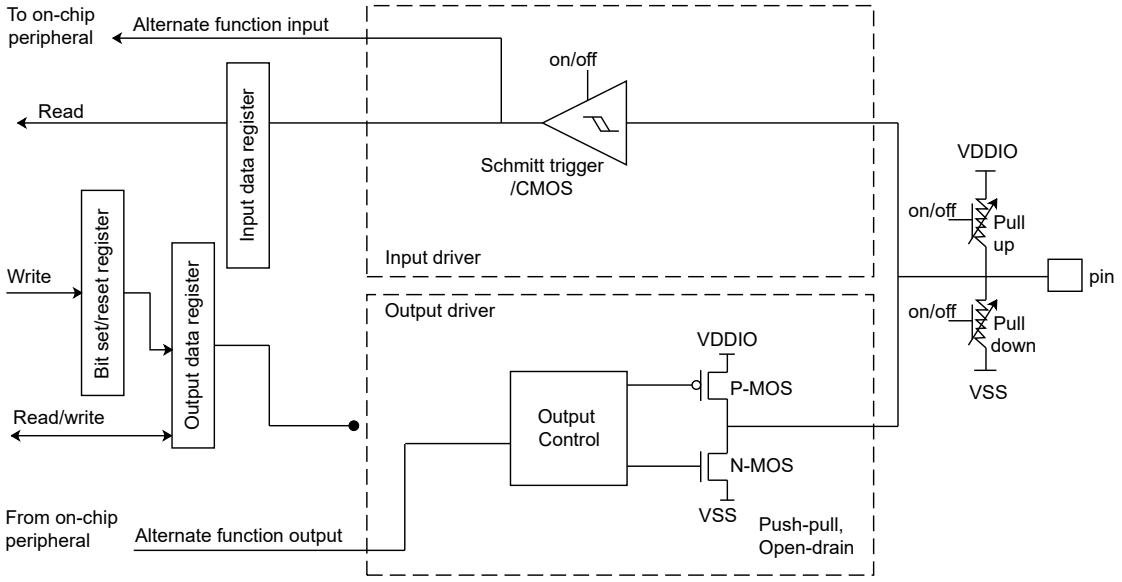


Figure 10-4 Alternate Function Configuration

10.3.12 Analog Configuration

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

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down is configurable with the GPIOx_PUPDR register.
- Read access to the input data register gets the value “0”.
- Analog input channel can be enabled or disabled by setting the GPIOx_ASR register in case the IOs have the Analog Switch function.

Figure 10-5 shows the analog-input configuration of the I/O port bits.

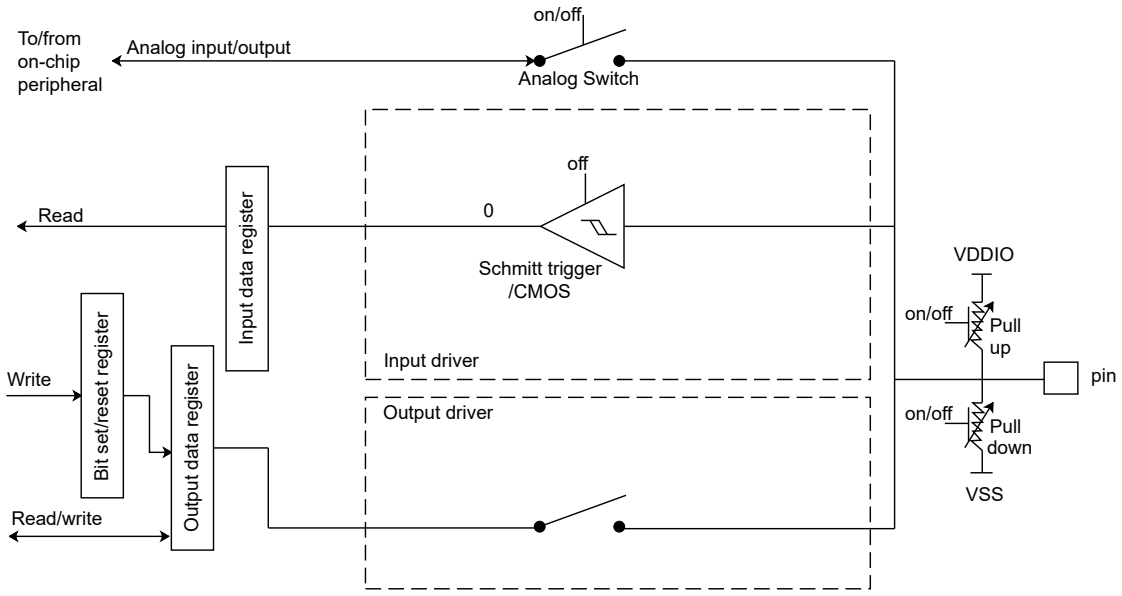


Figure 10-5 Analog Configuration

10.3.13 Using EHS or ELS Oscillator Pins as GPIOs

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

When the EHS or ELS oscillator is switched ON (by setting the EHSON in the RCC_CR register or ELSON bit in the RCC_BDCR register) the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the OSCH_IN or OSCL_IN pin is reserved for clock input and the OSCH_OUT or OSCL_OUT pin can still be used as normal GPIO.

10.3.14 Using GPIO Pins in the RTC Supply Domain

The PE7 and PE8 functionality is not activated when the core supply domain is powered off (when the device enters stop/standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to RTC description.

10.3.15 Using PE11 as GPIO

PE11 can be used as boot pin (BOOT) or as a GPIO. Depending on the nSWBOOT0 bit in the user option byte, it switches from the input mode to the analog input mode:

- It is loaded after the option byte if nSWBOOT0 = 1.
- It is after reset if nSWBOOT0 = 0.

10.3.16 GPIO with Deglitch Filter Control

All GPIOs can be enabled or disabled with the deglitch filter control function when the user wants to filter the deglitch impact on the GPIO signal, and it can be configured with the GPIOx_DFR (x=A..F) register.

10.4 Registers

10.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	GPIOx_MODER (x = A to F)	GPIO port mode register
0x0004	GPIOx_OTYPER (x = A to F)	GPIO port output type register
0x0008	GPIOx_OSPEEDR (x = A to F)	GPIO port speed rate register Refer to device datasheet for the frequency supported for different type GPIO.
0x000c	GPIOx_PUPDR (x = A to F)	GPIO port pull-up/pull-down register
0x0010	GPIOx_DRR (x = A to F)	GPIO port high driver register
0x0014	GPIOx_IDR (x = A to F)	GPIO port input data register
0x0018	GPIOx_ODR (x = A to F)	GPIO port output data register
0x001c	GPIOx_BSRR (x = A to F)	GPIO port bit set/reset register
0x0020	GPIOx_LCKR (x = A to F)	GPIO port configuration lock register
0x0024	GPIOx_AFRL (x = A to F)	GPIO alternate function low register

Offset	Register Name	Register Description
0x0028	GPIOx_AFRH (x = A to F)	GPIO alternate function high register
0x002c	GPIOx_BRR (x = A to F)	GPIO port bit reset register
0x0030	GPIOx_DFR (x = A to F)	GPIO port bit deglitch filter register
0x0034	GPIOx_ASR (x = A to F)	GPIO port bit analog switch register Refer to device datasheet for which port should use the analog switch.
0x0038	GPIOx_CSR (x = A to F)	GPIO port bit CMOS/schmitt trigger register

10.4.2 Register Field Details

10.4.2.1 GPIOx_MODER (x = A to F)

Address offset: 0x0000

Reset value: 0xFFFF FFFF (port A, B, C, D, F)

Reset value: 0xFFFF EAAF (port E)

0x0000			GPIO port mode register												GPIOx_MODER	
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	MODE15		MODE14		MODE13		MODE12		MODE11		MODE10		MODE9		MODE8	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MODE7		MODE6		MODE5		MODE4		MODE3		MODE2		MODE1		MODE0	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 10-2 GPIO Port Mode Register Description

Field	Name	Description
31:30	MODE15	Port15 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
29:28	MODE14	Port14 mode bits 00: Input mode

Field	Name	Description
		01: General purpose output mode 10: Alternate function mode 11: Analog mode
27:26	MODE13	Port13 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
25:24	MODE12	Port12 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
23:22	MODE11	Port11 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
21:20	MODE10	Port10 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
19:18	MODE9	Port9 mode bits 00: Input mode 01: General purpose output mode

Field	Name	Description
		10: Alternate function mode 11: Analog mode
17:16	MODE8	Port8 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
15:14	MODE7	Port7 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
13:12	MODE6	Port6 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
11:10	MODE5	Port5 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
9:8	MODE4	Port4 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode

Field	Name	Description
		11: Analog mode
7:6	MODE3	Port3 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
5:4	MODE2	Port2 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
3:2	MODE1	Port1 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode
1:0	MODE0	Port0 mode bits 00: Input mode 01: General purpose output mode 10: Alternate function mode 11: Analog mode

10.4.2.2 GPIOx_OTYPER (x = A to F)

Address offset: 0x0004

Reset value: 0x0000 0000

0x0004			GPIO port output type register											GPIOx_OTYPER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-3 GPIO Port Output Type Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	OT15	Port15 input data type 0: Output push-pull 1: Output open-drain
14	OT14	Port14 input data type 0: Output push-pull 1: Output open-drain
13	OT13	Port13 input data type 0: Output push-pull 1: Output open-drain

Field	Name	Description
12	OT12	Port12 input data type 0: Output push-pull 1: Output open-drain
11	OT11	Port11 input data type 0: Output push-pull 1: Output open-drain
10	OT10	Port10 input data type 0: Output push-pull 1: Output open-drain
9	OT9	Port9 input data type 0: Output push-pull 1: Output open-drain
8	OT8	Port8 input data type 0: Output push-pull 1: Output open-drain
7	OT7	Port7 input data type 0: Output push-pull 1: Output open-drain
6	OT6	Port6 input data type 0: Output push-pull 1: Output open-drain
5	OT5	Port5 input data type 0: Output push-pull 1: Output open-drain
4	OT4	Port4 input data type

Field	Name	Description
		0: Output push-pull 1: Output open-drain
3	OT3	Port3 input data type 0: Output push-pull 1: Output open-drain
2	OT2	Port2 input data type 0: Output push-pull 1: Output open-drain
1	OT1	Port1 input data type 0: Output push-pull 1: Output open-drain
0	OT0	Port0 input data type 0: Output push-pull 1: Output open-drain

10.4.2.3 GPIOx_OSPEEDR (x = A to F)

Address offset: 0x0008

Reset value: 0x0000 0000 (port A, B, C, D, F)

Reset value: 0x0000 0C00 (port E)

0x0008			GPIO port speed rate register											GPIOx_OSPEEDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	OSPEED15		OSPEED14		OSPEED13		OSPEED12		OSPEED11		OSPEED10		OSPEED9		OSPEED8	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OSPEED7		OSPEED6		OSPEED5		OSPEED4		OSPEED3		OSPEED2		OSPEED1		OSPEED0	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-4 GPIO Port Speed Rate Register Description

Field	Name	Description
31:30	OSPEED15	Port15 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
29:28	OSPEED14	Port14 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz

Field	Name	Description
		11: 120 MHz
27:26	OSPEED13	Port13 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
25:24	OSPEED12	Port12 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
23:22	OSPEED11	Port11 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
21:20	OSPEED10	Port10 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
19:18	OSPEED9	Port9 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz

Field	Name	Description
17:16	OSPEED8	Port8 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
15:14	OSPEED7	Port7 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
13:12	OSPEED6	Port6 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
11:10	OSPEED5	Port5 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
9:8	OSPEED4	Port4 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
7:6	OSPEED3	Port3 output speed

Field	Name	Description
		00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
5:4	OSPEED2	Port2 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
3:2	OSPEED1	Port1 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz
1:0	OSPEED0	Port0 output speed 00: 24 MHz 01: 48 MHz 10: 96 MHz 11: 120 MHz

10.4.2.4 GPIOx_PUPDR (x = A to F)

Address offset: 0x000c

Reset value: 0x0000 0000 (port A, B, C, D, F)

Reset value: 0x0000 2510 (port E)

0x000c			GPIO port pull-up/pull-down register											GPIOx_PUPDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PUPD15		PUPD14		PUPD13		PUPD12		PUPD11		PUPD10		PUPD9		PUPD8	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PUPD7		PUPD6		PUPD5		PUPD4		PUPD3		PUPD2		PUPD1		PUPD0	
Type	RW		RW		RW		RW		RW		RW		RW		RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-5 GPIO Port Pull-Up/Pull-Down Register Description

Field	Name	Description
31:30	PUPD15	Port15 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
29:28	PUPD14	Port14 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down

Field	Name	Description
		11: Reserved
27:26	PUPD13	Port13 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
25:24	PUPD12	Port12 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
23:22	PUPD11	Port11 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
21:20	PUPD10	Port10 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
19:18	PUPD9	Port9 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved

Field	Name	Description
17:16	PUPD8	Port8 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
15:14	PUPD7	Port7 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
13:12	PUPD6	Port6 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
11:10	PUPD5	Port5 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
9:8	PUPD4	Port4 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
7:6	PUPD3	Port3 pull-up/pull-down configuration bits

Field	Name	Description
		00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
5:4	PUPD2	Port2 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
3:2	PUPD1	Port1 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved
1:0	PUPD0	Port0 pull-up/pull-down configuration bits 00: No pull-up, pull-down 01: Pull-up 10: Pull-down 11: Reserved

10.4.2.5 GPIOx_DRR (x = A to F)

Address offset: 0x0010

Reset value: 0x0000 0000

0x0010			GPIO port high driver register											GPIOx_DRR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR15	DR14	DR13	DR12	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-6 GPIO Port High Driver Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	DR15	Port15 driving strength control 0: 10mA 1: 20mA
14	DR14	Port14 driving strength control 0: 10mA 1: 20mA
13	DR13	Port13 driving strength control 0: 10mA 1: 20mA

Field	Name	Description
12	DR12	Port12 driving strength control 0: 10mA 1: 20mA
11	DR11	Port11 driving strength control 0: 10mA 1: 20mA
10	DR10	Port10 driving strength control 0: 10mA 1: 20mA
9	DR9	Port9 driving strength control 0: 10mA 1: 20mA
8	DR8	Port8 driving strength control 0: 10mA 1: 20mA
7	DR7	Port7 driving strength control 0: 10mA 1: 20mA
6	DR6	Port6 driving strength control 0: 10mA 1: 20mA
5	DR5	Port5 driving strength control 0: 10mA 1: 20mA
4	DR4	Port4 driving strength control

Field	Name	Description
		0: 10mA 1: 20mA
3	DR3	Port3 driving strength control 0: 10mA 1: 20mA
2	DR2	Port2 driving strength control 0: 10mA 1: 20mA
1	DR1	Port1 driving strength control 0: 10mA 1: 20mA
0	DR0	Port0 driving strength control 0: 10mA 1: 20mA

10.4.2.6 GPIOx_IDR (x = A to F)

Address offset: 0x0014

Reset value: 0x0000xxxx

0x0014			GPIO Port Input Data Register											GPIOx_IDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-7 GPIO Port Input Data Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	IDR15	Port15 input data
14	IDR14	Port14 input data
13	IDR13	Port13 input data
12	IDR12	Port12 input data
11	IDR11	Port11 input data
10	IDR10	Port10 input data
9	IDR9	Port9 input data

Field	Name	Description
8	IDR8	Port8 input data
7	IDR7	Port7 input data
6	IDR6	Port6 input data
5	IDR5	Port5 input data
4	IDR4	Port4 input data
3	IDR3	Port3 input data
2	IDR2	Port2 input data
1	IDR1	Port1 input data
0	IDR0	Port0 input data

10.4.2.7 GPIOx_ODR (x = A to F)

Address offset: 0x0018

Reset value: 0x0000 0000

0x0018			GPIO port output data register											GPIOx_ODR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-8 GPIO Port Output Data Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	ODR15	Port15 output data
14	ODR14	Port14 output data
13	ODR13	Port13 output data
12	ODR12	Port12 output data
11	ODR11	Port11 output data
10	ODR10	Port10 output data
9	ODR9	Port9 output data

Field	Name	Description
8	ODR8	Port8 output data
7	ODR7	Port7 output data
6	ODR6	Port6 output data
5	ODR5	Port5 output data
4	ODR4	Port4 output data
3	ODR3	Port3 output data
2	ODR2	Port2 output data
1	ODR1	Port1 output data
0	ODR0	Port0 output data

10.4.2.8 GPIOx_BSRR (x = A to F)

Address offset: 0x001c

Reset value: 0x0000 0000

0x001c			GPIO port bit set/reset register											GPIOx_BSRR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-9 GPIO Port Bit Set/Reset Register Description

Field	Name	Description
31	BR15	Port15 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
30	BR14	Port14 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
29	BR13	Port13 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
28	BR12	Port12 reset bit

Field	Name	Description
		0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
27	BR11	Port11 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
26	BR10	Port10 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
25	BR9	Port9 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
24	BR8	Port8 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
23	BR7	Port7 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
22	BR6	Port6 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
21	BR5	Port5 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
20	BR4	Port4 reset bit 0: No action on the corresponding ODRx bit

Field	Name	Description
		1: Reset the corresponding ODRx bit
19	BR3	Port3 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
18	BR2	Port2 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
17	BR1	Port1 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
16	BR0	Port0 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
15	BS15	Port15 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
14	BS14	Port14 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
13	BS13	Port13 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
12	BS12	Port12 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit

Field	Name	Description
11	BS11	Port11 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
10	BS10	Port10 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
9	BS9	Port9 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
8	BS8	Port8 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
7	BS7	Port7 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
6	BS6	Port6 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
5	BS5	Port5 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
4	BS4	Port4 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
3	BS3	Port3 set bit:

Field	Name	Description
		0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
2	BS2	Port2 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
1	BS1	Port1 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit
0	BS0	Port0 set bit: 0: No action on the corresponding ODRx bit 1: Set the corresponding ODRx bit

10.4.2.9 GPIOx_LCKR (x = A to F)

Address offset: 0x0020

Reset value: 0x0000 0000

0x0020			GPIO port configuration lock register											GPIOx_LCKR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															LCKK
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-10 GPIO Port Configuration Lock Register Description

Field	Name	Description
31:17	Reserved	Reserved
16	LCKK	Lock key 0: Port configuration lock key not active 1: Port configuration lock key active LOCK key writing sequence: W 1 - W 0 - W 1 - R 0 - R 1 (This read is the option to confirm the lock is active.)
15	LCK15	Port15 lock bit 0: Port configuration not locked 1: Port configuration locked
14	LCK14	Port14 lock bit

Field	Name	Description
		0: Port configuration not locked 1: Port configuration locked
13	LCK13	Port13 lock bit 0: Port configuration not locked 1: Port configuration locked
12	LCK12	Port12 lock bit 0: Port configuration not locked 1: Port configuration locked
11	LCK11	Port11 lock bit 0: Port configuration not locked 1: Port configuration locked
10	LCK10	Port10 lock bit 0: Port configuration not locked 1: Port configuration locked
9	LCK9	Port9 lock bit 0: Port configuration not locked 1: Port configuration locked
8	LCK8	Port8 lock bit 0: Port configuration not locked 1: Port configuration locked
7	LCK7	Port7 lock bit 0: Port configuration not locked 1: Port configuration locked
6	LCK6	Port6 lock bit 0: Port configuration not locked

Field	Name	Description
		1: Port configuration locked
5	LCK5	Port5 lock bit 0: Port configuration not locked 1: Port configuration locked
4	LCK4	Port4 lock bit 0: Port configuration not locked 1: Port configuration locked
3	LCK3	Port3 lock bit 0: Port configuration not locked 1: Port configuration locked
2	LCK2	Port2 lock bit 0: Port configuration not locked 1: Port configuration locked
1	LCK1	Port1 lock bit 0: Port configuration not locked 1: Port configuration locked
0	LCK0	Port0 lock bit 0: Port configuration not locked 1: Port configuration locked

10.4.2.10 GPIOx_AFRL (x = A to F)

Address offset: 0x0024

Reset value: 0x0000 0000

0x0024			GPIO alternate function low register											GPIOx_AFRL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	AF7				AF6				AF5				AF4			
Type	RW				RW				RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	AF3				AF2				AF1				AF0			
Type	RW				RW				RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-11 GPIO Alternate Function Low Register Description

Field	Name	Description
31:28	AF7	Alternate function selection for port 7 pin
27:24	AF6	Alternate function selection for port 6 pin
23:20	AF5	Alternate function selection for port 5 pin
19:16	AF4	Alternate function selection for port 4 pin
15:12	AF3	Alternate function selection for port 3 pin
11:8	AF2	Alternate function selection for port 2 pin
7:4	AF1	Alternate function selection for port 1 pin
3:0	AF0	Alternate function selection for port 0 pin

10.4.2.11 GPIOx_AFRH (x = A to F)

Address offset: 0x0028

Reset value: 0x0000 0000

0x0028			GPIO alternate function high register											GPIOx_AFRH		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	AF15				AF14				AF13				AF12			
Type	RW				RW				RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	AF11				AF10				AF9				AF8			
Type	RW				RW				RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-12 GPIO Alternate Function High Register Description

Field	Name	Description
31:28	AF15	Alternate function selection for port 15 pin
27:24	AF14	Alternate function selection for port 14 pin
23:20	AF13	Alternate function selection for port 13 pin
19:16	AF12	Alternate function selection for port 12 pin
15:12	AF11	Alternate function selection for port 11 pin
11:8	AF10	Alternate function selection for port 10 pin
7:4	AF9	Alternate function selection for port 9 pin
3:0	AF8	Alternate function selection for port 8 pin

10.4.2.12 GPIOx_BRR (x = A to F)

Address offset: 0x002c

Reset value: 0x0000 0000

0x002c			GPIO port bit reset register											GPIOx_BRR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-13 GPIO Port Bit Reset Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	BR15	Port15 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
14	BR14	Port14 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
13	BR13	Port13 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit

Field	Name	Description
12	BR12	Port12 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
11	BR11	Port11 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
10	BR10	Port10 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
9	BR9	Port9 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
8	BR8	Port8 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
7	BR7	Port7 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
6	BR6	Port6 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
5	BR5	Port5 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
4	BR4	Port4 reset bit

Field	Name	Description
		0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
3	BR3	Port3 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
2	BR2	Port2 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
1	BR1	Port1 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit
0	BR0	Port0 reset bit 0: No action on the corresponding ODRx bit 1: Reset the corresponding ODRx bit

10.4.2.13 GPIOx_DFR (x = A to F)

Address offset: 0x0030

Reset value: 0xFFFF FFFF

0x0030			GPIO port bit deglitch filter register											GPIOx_DFR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DF15	DF14	DF13	DF12	DF11	DF10	DF9	DF8	DF7	DF6	DF5	DF4	DF3	DF2	DF1	DF0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-14 GPIO Port Bit Deglitch Filter Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	DF15	Port15 deglitch filter 0: With deglitch filter 1: Without deglitch filter
14	DF14	Port14 deglitch filter 0: With deglitch filter 1: Without deglitch filter
13	DF13	Port13 deglitch filter 0: With deglitch filter 1: Without deglitch filter

Field	Name	Description
12	DF12	Port12 deglitch filter 0: With deglitch filter 1: Without deglitch filter
11	DF11	Port11 deglitch filter 0: With deglitch filter 1: Without deglitch filter
10	DF10	Port10 deglitch filter 0: With deglitch filter 1: Without deglitch filter
9	DF9	Port9 deglitch filter 0: With deglitch filter 1: Without deglitch filter
8	DF8	Port8 deglitch filter 0: With deglitch filter 1: Without deglitch filter
7	DF7	Port7 deglitch filter 0: With deglitch filter 1: Without deglitch filter
6	DF6	Port6 deglitch filter 0: With deglitch filter 1: Without deglitch filter
5	DF5	Port5 deglitch filter 0: With deglitch filter 1: Without deglitch filter
4	DF4	Port4 deglitch filter

Field	Name	Description
		0: With deglitch filter 1: Without deglitch filter
3	DF3	Port3 deglitch filter 0: With deglitch filter 1: Without deglitch filter
2	DF2	Port2 deglitch filter 0: With deglitch filter 1: Without deglitch filter
1	DF1	Port1 deglitch filter 0: With deglitch filter 1: Without deglitch filter
0	DF0	Port0 deglitch filter 0: With deglitch filter 1: Without deglitch filter

10.4.2.14 GPIOx_ASR (x = A to F)

Address offset: 0x0034

Reset value: 0x0000 0000

0x0034			GPIO port bit analog switch register											GPIOx_ASR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	AS15	AS14	AS13	AS12	AS11	AS10	AS9	AS8	AS7	AS6	AS5	AS4	AS3	AS2	AS1	AS0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-15 GPIO Port Bit Analog Switch Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	AS15	Port15 analog switch 0: Disable 1: Enable
14	AS14	Port14 analog switch 0: Disable 1: Enable
13	AS13	Port13 analog switch 0: Disable 1: Enable

Field	Name	Description
12	AS12	Port12 analog switch 0: Disable 1: Enable
11	AS11	Port11 analog switch 0: Disable 1: Enable
10	AS10	Port10 analog switch 0: Disable 1: Enable
9	AS9	Port9 analog switch 0: Disable 1: Enable
8	AS8	Port8 analog switch 0: Disable 1: Enable
7	AS7	Port7 analog switch 0: Disable 1: Enable
6	AS6	Port6 analog switch 0: Disable 1: Enable
5	AS5	Port5 analog switch 0: Disable 1: Enable
4	AS4	Port4 analog switch

Field	Name	Description
		0: Disable 1: Enable
3	AS3	Port3 analog switch 0: Disable 1: Enable
2	AS2	Port2 analog switch 0: Disable 1: Enable
1	AS1	Port1 analog switch 0: Disable 1: Enable
0	AS0	Port0 analog switch 0: Disable 1: Enable

10.4.2.15 GPIOx_CSR (x = A to F)

Address offset: 0x0038

Reset value: 0x0000 0000

0x0038			GPIO port bit CMOS/schmitt trigger register											GPIOx_CSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CS15	CS14	CS13	CS12	CS11	CS10	CS9	CS8	CS7	CS6	CS5	CS4	CS3	CS2	CS1	CS0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-16 GPIO Port Bit CMOS/Schmitt Trigger Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	CS15	Port15 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
14	CS14	Port14 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
13	CS13	Port13 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer

Field	Name	Description
12	CS12	Port12 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
11	CS11	Port11 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
10	CS10	Port10 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
9	CS9	Port9 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
8	CS8	Port8 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
7	CS7	Port7 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
6	CS6	Port6 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
5	CS5	Port5 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
4	CS4	Port4 CMOS/Schmitt trigger input buffer configuration

Field	Name	Description
		0: Schmitt trigger input buffer 1: CMOS input buffer
3	CS3	Port3 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
2	CS2	Port2 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
1	CS1	Port1 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer
0	CS0	Port0 CMOS/Schmitt trigger input buffer configuration 0: Schmitt trigger input buffer 1: CMOS input buffer

Quad Serial Peripheral Interface (QSPI)

This chapter describes the details of the Quad Serial Peripheral Interface (QSPI).

Topics: **Page**

11.1 Introduction

The Quad Serial Peripheral Interface (QSPI) Controller Unit is part of ARM China Star Processor and provides a mechanism to communicate with the SPI Flash memory device. QSPI provides the necessary functionality for a host to communicate with a serial Flash device through the SPI. The unit supports almost common serial Flash device instructions, such as read, program, erase and other custom instructions. QSPI provides a mechanism to load executing programs directly from external Flash memory instead of boot-up from embedding Flash memory.

11.2 Features

- Support for single, dual, and quad I/O commands
- Direct access, indirect access, and inactive modes
- Execution in place (XIP) mode support
- Support for Single Data Rate (SDR) and Double Data Rate (DDR)
- Integrated FIFO for reception and transmission (the depth of FIFO is 8 bytes)
- 8-bit, 16-bit, and 32-bit data accesses are allowed in indirect mode
- Interrupt free
- Single operation clock and single asynchronous reset

11.3 QSPI Controller Unit

The QSPI provides the necessary functionality to a host to communicate with a serial Flash device through the SPI. The unit supports most common serial Flash device instructions, such as read, program, erase and other custom instructions.

The communication with Flash devices is used by commands, which include five phases: Instruction, Address, Alternate byte, Dummy and Data. Any of these phases can be configured to be skipped, but at least one of them needs to be present.

Figure 11-1 shows an example of a read transaction in quad SPI mode.

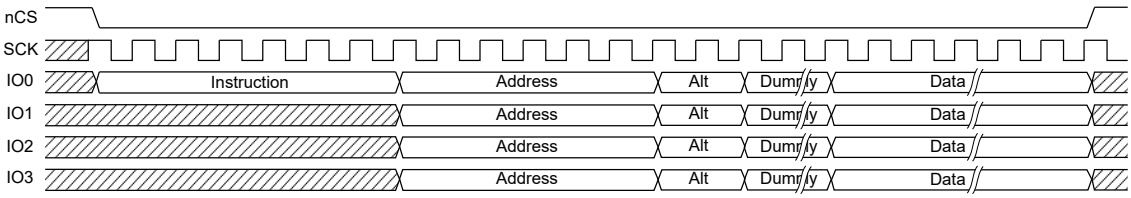


Figure 11-1 Example of a Read Transaction in Quad SPI Mode

11.3.1 Instruction Phase

This is an 8-bit instruction, with its value set through the INSTRUCTION bit in either QSPI_RMCR or QSPI_OMCR, depending on the current operation mode defined by the OPMODE bit in QSPI_CR. The number of lines used for transmitting the instruction phase, or its absence, is dictated by the IMODE bit in either QSPI_RMCR or QSPI_OMCR.

11.3.2 Address Phase

The number of address bytes, ranging from 1 to 4, is configured in the ADSIZE bit of either QSPI_RMCR or QSPI_OMCR. The ADMODE bit in these same registers determines the line count used for sending address phase or its absence. In indirect mode, when this phase is included, the address value must also be configured in QSPI_IMAR. Conversely, in direct mode, the address is supplied directly through AHB.

11.3.3 Alternate-bytes Phase

In the alternate-bytes phase, 1-4 bytes are sent to the Flash memory, typically to control the mode of operation. The number of alternate bytes to be sent is configured via the ABSIZE bit in either the QSPI_RMCR or QSPI_OMCR. The content to be sent is specified in QSPI_RABR and QSPI_OABR respectively for each mode. The ABMODE bit, located in QSPI_RMCR or QSPI_OMCR, decides the number of lines to be used to send alternate-bytes phase or its absence.

11.3.4 Dummy-cycles Phase

In the dummy-cycles phase, 0-31 dummy cycles are added to Flash memory, generally to allow the Flash device to prepare the data phase. The number of dummy cycles to be sent is configured via the NUMDC bit in either QSPI_RMCR or QSPI_OMCR. Its unit is a full SCK cycle.

11.3.5 Data Phase

In the data phase, any number of bytes can be transferred to or from the Flash memory. The data size can be determined by the DSIZE bit in either QSPI_RMCR (which only supports 32 bits in direct mode) or QSPI_OMCR. The number of using lines in the data phase is configured in the DMODE bit in QSPI_RMCR or QSPI_OMCR.

In indirect mode, if the DMODE bit in QSPI_OMCR is set to 0, the data phase is skipped. If not, the data is sent or received by FIFO. It's necessary to configure the QSPI_DLR with the required data length and then read or write the data through the QSPI_FDR. The FIFO status can be obtained from the QSPI_SR.

QSPI is a specialized communication interface targeting single, dual or quad SPI Flash memories. It can operate in any of the three following modes:

- **Direct Read Access 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. Direct read access mode can be used to both access and directly execute code from external Flash memory, and it supports Flash memory XIP mode. After power-on, QSPI changes to the default direct read access mode to boot-up. The operation mode is accessed through AHB-Bus. When in the direct read mode, any other modes can be inserted at any time.

- **Indirect Mode**

All the operations are performed using the registers. It allows the software to access the internal TX FIFO and RX FIFO directly. The level of FIFO is 8 bytes. It is used to access the volatile and non-volatile configuration registers, the legacy SPI status registers, other status and protection registers and the Flash ROM content. It is recommended that this mode is used to erase and configure the serial Flash device. When a transaction request is from Q-AHB, the transaction will be waiting until exiting the current Indirect mode into direct mode.

- **Inactive Mode**

This mode is used to disable QSPI-Bus. It offers a ‘clean’ state to set up the QSPI’s related register, such as division clock index, command mode, command type, and so on. When a transaction request is from Q-AHB, the transaction will be an error response.

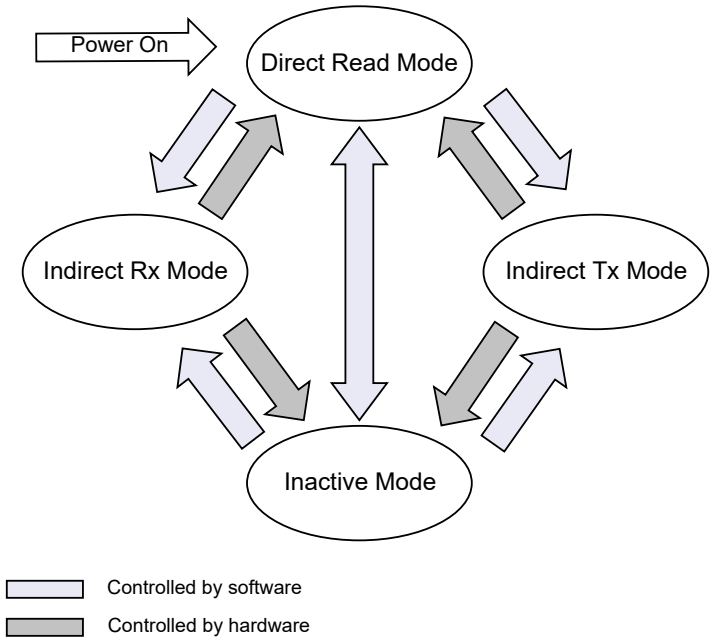


Figure 11-2 QSPI State Flowchart

The initial state is direct read mode after power on or cold reset. You can configure QSPI to other states by setting the OPMODE bit in QSPI_CR. Indirect mode is a special state. When the transaction in indirect mode is done, hardware will be responsible for changing the state back to the previous one (Inactive or Direct Read, depending on the mode which it enters from). The TCF bit in QSPI_SR is used to indicate whether the indirect transaction is done.

Programmers can control each mode's behavior via corresponding registers as follows:

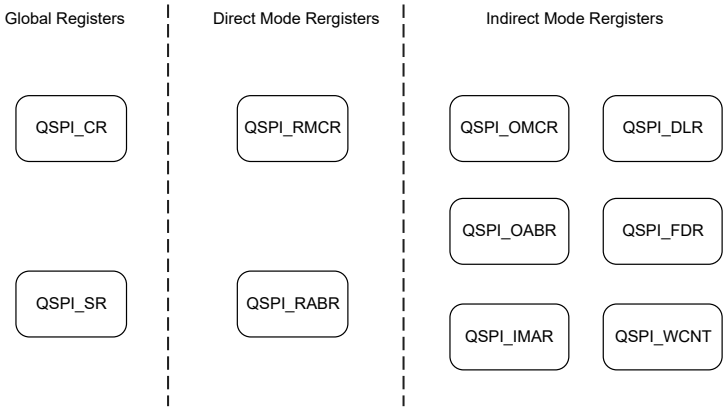


Figure 11-3 QSPI Register List

The direct read mode and indirect mode use different sets of registers to construct respective SPI communication. These settings take effect only after the QSPI_CR register is configured, which means that you must confirm the target mode, direct or indirect mode, then write the corresponding registers as shown in Figure 11-3. When the QSPI_CR is configured, the QSPI will load mode-specific parameters based on the value of the OPMODE bit in the QSPI_CR.

11.3.6 Direct Read Mode Operation

It is the default state after QSPI power-on or cold reset, the external SPI Flash device can be seen as an internal memory. To read the Flash memory normally, it's crucial to ensure that QSPI can access the memory with its default settings. If not, a reconfiguration of QSPI is necessary.

No more than 256 MB can be addressed even if the Flash memory capacity is larger. The memory size can be configured via the FMSIZE bit in the QSPI_CR. Any attempts to access an address beyond this register's definition will trigger a bus error.

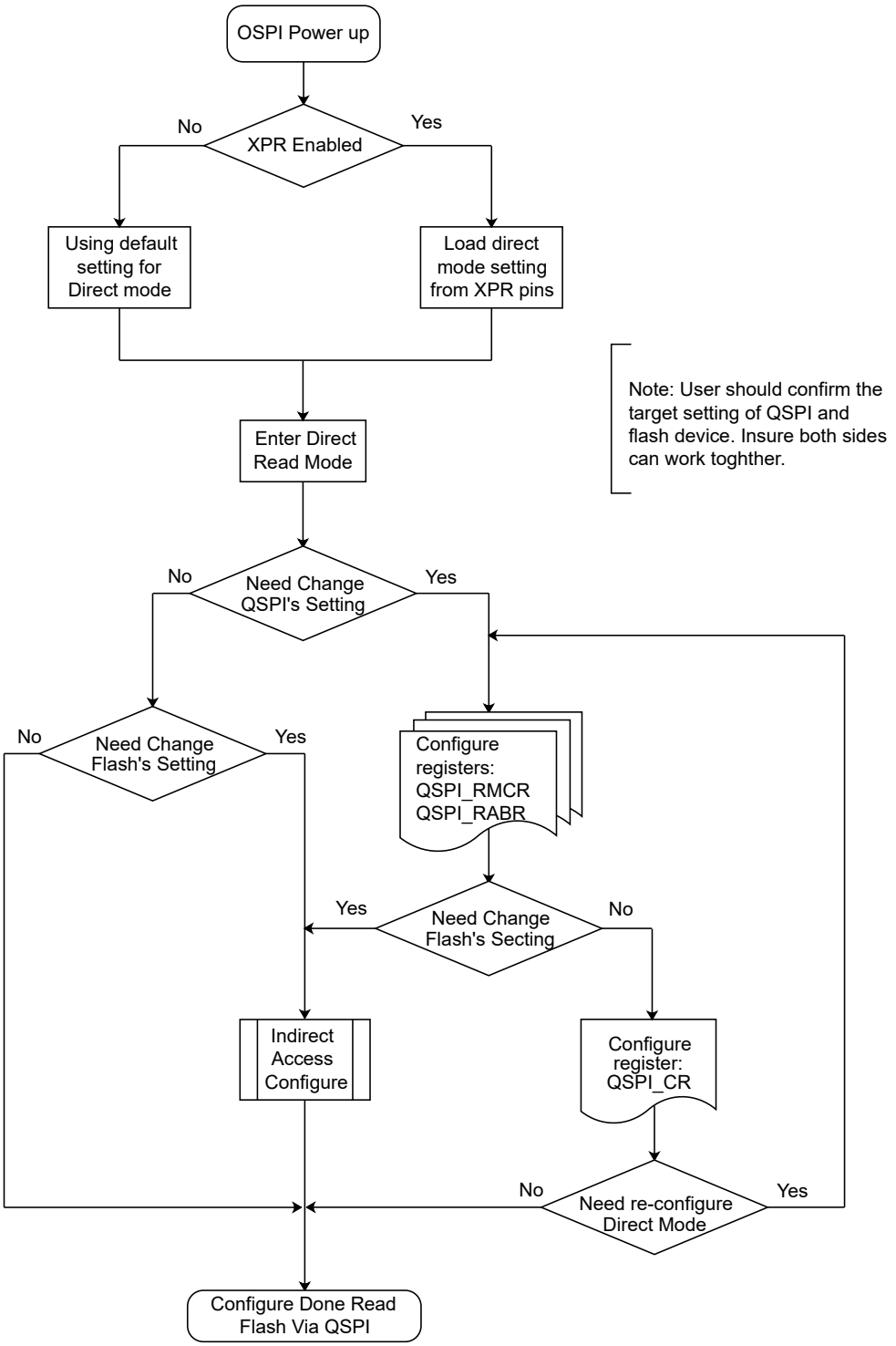
11.3.7 Indirect Mode

The QSPI enters indirect mode by writing the OPMODE bit in QSPI_CR to 2'b10. The indirect mode registers will be used to construct the communication protocol. If the DMODE bit in the QSPI_OMCR is not 0, the QSPI will send or receive a certain number of data via the FIFO register QSPI_FDR. The data's length is configured by QSPI_DLR. The data's direction is determined by the IDMODE bit in the QSPI_OMCR. When the programmed number of bytes to be sent or received is reached, the TCF bit in QSPI_SR is set.

11.3.8 Inactive Mode

When the OPMODE bit in the QSPI_CR is set to 2'b11, QSPI enters an inactive mode, where any direct access to the Flash memory results in a bus error. This mode can also serve as an initial state when launching an indirect transaction, but note that it will return to inactive mode upon finishing transmission.

11.3.9 Program Flow



NOTE: XPR is not supported on QSPI controller implemented on this device.

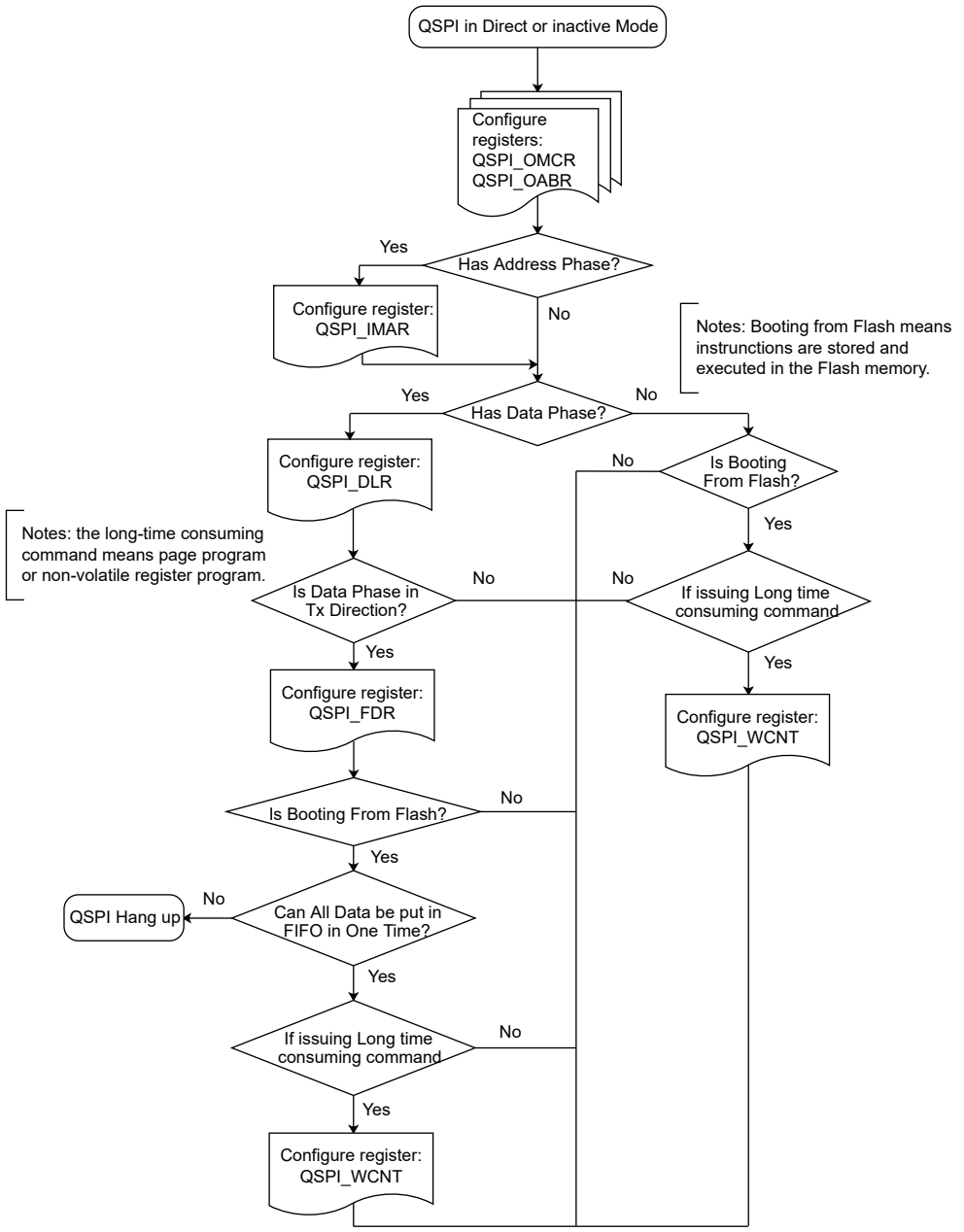


Figure 11-4 QSPI Program Flow 1

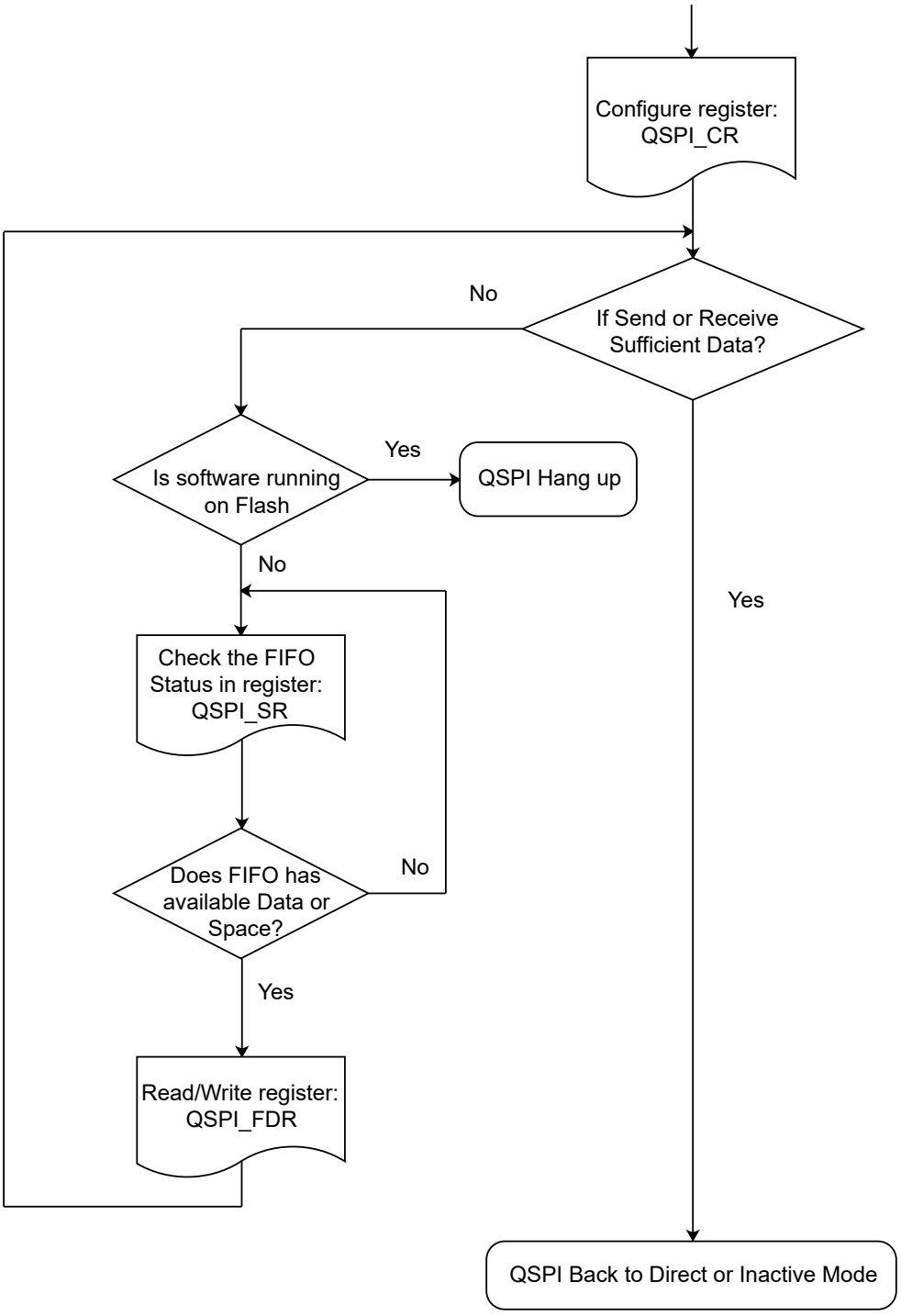


Figure 11-5 QSPI Program Flow 2

Direct Read Operation Setup

1. Wait until any pending operation is complete.
2. Update the IMODE bit in QSPI_RMCR with the instruction mode (transmit instruction on single/dual/quad lines) that you want to use for direct read mode.
3. If the IMODE bit in QSPI_RMCR is not zero, update the INSTRUCTION bit in QSPI_RMCR with the instruction ID that you want to use in direct read mode.
4. Update the bits ADMODE, ADSIZE, ABMODE, ABSIZE, DMODE and NUMDC in QSPI_RMCR, respectively, if any of them exist in your coming direct read operation.

5. Update the RXDLY bit in QSPI_RMCR as needed. It specifies the number of delay steps for sampling the input serial data. The unit is core running clock cycle.
6. Update the WRAPMODE and NOWRAP bits in QSPI_RMCR as needed.
7. Update the DDRMODE bit in QSPI_RMCR as you want to use.
8. Update QSPI_RABR with the expected value if the ABMODE bit in QSPI_RMCR is not zero.
9. Launch an indirect access operation to configure the Flash to ensure that it is in correct state as needed when software is running in Flash memory.
10. Update the OPMODE bit in QSPI_CR in direct read mode and other bits as needed, then write the combined value to the QSPI_CR. Any write access to this register will be considered as a sign to update the corresponding mode's settings.
11. You can read Flash memory directly like normal memory.

NOTE: For step 9, if software is running on Flash memory, you need to keep both QSPI and Flash memory work in the same SPI protocol after changing to Direct Read Mode, otherwise the CPU cannot fetch the instruction from Flash, and the system will hang up.

Indirect Access Mode Setup

1. Make sure that QSPI is in direct read or inactive mode.
2. Update the IMODE bit in QSPI_OMCR with the instruction mode (transmit instruction on single/dual/quad lines) that you want to use for indirect access mode.
3. If the IMODE bit in QSPI_OMCR is not zero, update the INSTRUCTION bit in QSPI_OMCR with the instruction ID you want to use in indirect access mode.
4. Update the bits ADMODE, ADSIZE, ABMODE, ABSIZE, DMODE, DSIZE and NUMDC in QSPI_OMCR, respectively, if any of them exist in your coming indirect access operation.
5. Update the IDMODE bit in QSPI_OMCR depending on the data phase direction, Rx or Tx.
6. Update the RXDLY bit in QSPI_OMCR as needed. It specifies the number of delay steps for sampling the input serial data. The unit is core clock cycle.
7. Update the WRAPMODE bit in QSPI_RMCR and NOWRAP bit in QSPI_RMCR as needed.
8. Update the DDRMODE bit in QSPI_OMCR as you want to use.
9. Update QSPI_OABR with the expected value if the ABMODE bit in QSPI_OMCR is not zero.
10. Update QSPI_IMAR with the expected value if the ADMODE bit in QSPI_OMCR is not zero.
11. Update QSPI_DLR with the expected data length (total bytes -1) if the DMODE bit in QSPI_OMCR is not zero. If it is Tx direction, you can fill the data in QSPI_FDR until the FIFO is full.
12. Update the OPMODE bit in QSPI_CR in indirect access mode and other fields as needed, then write the combined value to the QSPI_CR register. Any write access to this register will be considered as a sign to update the corresponding mode's settings.
13. Read or write (depending on the data phase direction in IDMODE bit in QSPI_OMCR) the QSPI_FDR register until it reaches to the expected data length, which is configured in QSPI_DLR.
14. QSPI returns to the previous state after the transaction finishes.

NOTE: For step 11, if software is running on Flash memory, you must fill QSPI_FDR with all the sending data before configuring QSPI_CR, so that the MAX length of data in this scenario is the FIFO depth. If software is not running on Flash memory, it is optional to write QSPI_FDR in this step. For step 12, if software is running on Flash memory, the writing QSPI_FDR action should not occur in this step.

XIP Mode Enter/Exit

XIP mode (some manufacturers call it set mode) is supported in most SPI Flash devices. However, Flash manufacturers do not use the consistent standard approach to enter/exit this mode. Some use the special value in alternate byte phase to indicate that the next continuous transaction has no Instruction phase. Some use the configuration register to enable XIP mode.

Entering XIP Mode via Alternate Byte Phase

1. Perform the steps as described in Direct Read Operation Setup, but set QSPI_RABR with the special value, which will inform Flash to enter XIP mode, and set the XIPMODE bit in QSPI_CR to 1.
2. QSPI and Flash will communicate in XIP mode in the next new transaction.

Entering XIP Mode via the Configuration Register

1. Prepare the direct read mode's setting, which is used in XIP mode via QSPI_RMCR and QSPI_RABR registers as needed, but without setting the QSPI_CR.
2. Perform the steps as described in Indirect Access Mode Setup, but set the XIPIMMENTER bit in QSPI_OMCR to 1.
3. After the indirect access transaction finishes, QSPI and Flash will both work in XIP mode immediately. The XIPSTATUS bit in QSPI_SR will indicate the QSPI status.

Exiting XIP Mode via Alternate Byte Phase

1. Perform the steps as described in Direct Read Operation Setup, but set QSPI_RABR with any value except the specific XIP one, which will inform Flash to exit XIP mode, and set the XIPMODE bit in QSPI_CR to 0.
2. QSPI and Flash will communicate in normal mode in the next new transaction.

Exiting XIP Mode via the Configuration Register

1. Prepare the direct read mode's setting, which is used in normal mode via registers QSPI_RMCR and QSPI_RABR as needed, but without setting the QSPI_CR register.
2. Perform the steps as described in Indirect Access Mode Setup, but set the XIPIMMENTER bit in QSPI_OMCR to 0.
3. After the indirect access transaction finishes, QSPI and Flash will both work in normal mode immediately. The XIPSTATUS bit in QSPI_SR will indicate the QSPI status.

11.3.10 Usage Examples

Set both QSPI and Flash memory from legacy SPI mode to Quad SPI mode

Initial state:

QSPI is in legacy SPI mode (single line, read instruction 0x3), and Flash memory only accepts instructions on a single line.

Target state:

Both QSPI and Flash memory should be in (Quad SPI mode, read instruction 0xB).

Program procedure:

1. According to the Flash memory datasheet, find the corresponding configuration for high-speed read instruction 0xB, and configure QSPI_RMCR to match it. If the alternate bytes phase is present, configure QSPI_RABR as well.
2. Prepare the Quad SPI enable command for Flash memory. Some Flash devices need a special command to enable Quad SPI, and the communication IO number will extend to 4 after this

command. Suppose that this instruction is 0x38. So, configure the QSPI_OMCR register to prepare this command, and be aware that it is still transmitted with the legacy SPI protocol.

3. Write QSPI_CR with OPMODE = Indirect Mode. QSPI will issue the Quad-SPI enabled command to Flash memory with the settings as described in step 2.
4. If QSPI's initial state is inactive, wait for the TCF flag in QSPI_SR, then write QSPI_CR with OPMODE = Direct Mode.
5. QSPI should be in direct mode, and the configuration in step 1 also takes effect.
6. QSPI will read Flash in direct mode with the Quad SPI protocol.

Set both QSPI and Flash memory into XIP mode

Initial state:

QSPI is in Quad SPI mode (Quad lines, Direct Read Mode).

Target state:

Both QSPI and Flash memory should be in (Quad SPI mode, XIP-enabled, Direct Read Mode).

XIP mode:

It means that there is no Instruction phase in command. Flash memory names it XIP mode or SET mode.

Program procedure:

1. According to the Flash memory datasheet, find the proper way to enter XIP mode. Normally, there are two ways for most Flash chips.
   a. Set the alternate-bytes phase with a special value. When Flash memory receives the first command which has such special Alternate-byte phase, it will expect the next continuous instruction to be another same Read command and does not require the Instruction phase.
   b. Use the special XIP-enabled command to enable the XIP mode immediately.
2. For Flash chips which need way a) to enter XIP mode:
   a. Configure QSPI_RABR with the special Alternate-bytes value.
   b. Write the QSPI_CR register with fields XIPMODE = 1 and OPMODE = 0.
   c. After one direct read transaction, both sides will be in XIP mode, which can be checked via the XIPSTATUS bit in QSPI_SR.
3. For Flash chips which need way b) to enter XIP mode:
   a. Prepare the XIP-enabled command in indirect registers. The XIPIMMENTER bit in QSPI_OMCR should be set.
   b. If necessary, prepare the Read command in direct registers, which will be used during XIP mode.
   c. Write the QSPI_CR register with OPMODE = 2'b10.
   d. When QSPI finishes the indirect transaction, it will load the direct registers setting and enter XIP mode immediately, which means that the next Direct Read transaction has no Instruction phase.

Page program Flash memory

Initial state:

QSPI is in Quad SPI mode (Quad lines, Direct Read Mode).

Target state:

Insert the Indirect Write transaction in direct mode.

Program procedure:

1. Prepare the Write-Enable (for instance, instruction:0x06) command in indirect register.
2. Write the OPMODE bit in QSPI_CR = 2'b10 to launch an indirect transaction.
3. Prepare the Erase-Memory (for instance, instruction:0x20) command in indirect register.
4. Write the OPMODE bit in QSPI_CR = 2'b10 to launch an indirect transaction.
5. Prepare the Write-Enable (for instance, instruction:0x06) command in indirect register.
6. Write the OPMODE bit in QSPI_CR = 2'b10 to launch an indirect transaction.
7. Prepare the page-program (for instance, instruction:0x02) command in indirect register, and the target programming address should be erased in the previous step. Ensure the DMODE bit in QSPI_OMCR is non-zero and that the value of (QSPI_DLR+1) matches the length of the programming data.
8. Fill QSPI_FDR with programming data until it is full. If software is running on Flash memory, the programming data length must not be larger than the FIFO depth, and you also need to configure the QSPI_WCNT register to a certain value. The AHB bus will be held for such number of clock, then return back to direct mode, which means that Flash memory needs to finish the page-programming action during this time. If the time is not long enough for Flash to finish it, QSPI cannot access the memory to fetch the code when it returns to direct mode. The software will hang up there.
9. If there is unsent data, check the FIFO status by examining the FIFOFULL bit in QSPI_SR, and continue filling QSPI_FDR with the unsent data until the expected data length is reached.
10. Check the TCF bit in QSPI_SR to confirm that the indirect transfer is complete.
11. QSPI will return to the previous mode.

NOTE: For page-programing or any other non-volatile register setting action, it is recommended to run on RAM, instead of Flash memory. If you insist on running this part of software on Flash memory, ensure that the waiting time (QSPI_WCNT) is enough for Flash to finish the operation and the sending data length does not exceed the FIFO depth.

11.3.11 Error Response Condition

In the following exception conditions, the bus will issue an error response.

- **AHB**
  - Read access in inactive mode.
  - Read access when FSM jumps back to inactive mode from indirect mode.
  - Read access's address fails in boundary check.

Any write accesses.

- **APB - Access FIFO must check the FIFO status first**
  - Read FIFO (addr: 0x1c) when the FIFO is empty.
  - Write FIFO (addr: 0x1c) when the FIFO is full.
  - Any write to the QSPI_CR register when the indirect access operation is not done or the previous operation mode change request is not granted.

11.4 Registers

11.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	QSPI_CR	QSPI control register
0x0004	QSPI_SR	QSPI status register
0x0008	QSPI_RMCR	QSPI direct read access mode control register
0x000c	QSPI_OMCR	QSPI indirect mode control register
0x0010	QSPI_RABR	QSPI direct eead mode alternate bytes register
0x0014	QSPI_OABR	QSPI indirect mode alternate bytes register
0x0018	QSPI_IMAR	QSPI indirect mode address register
0x001c	QSPI_FDR	QSPI FIFO data register
0x0020	QSPI_DLR	QSPI indirect mode data length register
0x0024	QSPI_WCNT	QSPI wait counter register

11.4.2 Register Field Details

11.4.2.1 QSPI_CR

0x0000			QSPI control register											QSPI_CR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		SCKSCALER						Reserved					CSRHT		
Type	RO		RW						RO					RW		
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		FMSIZE						Reserved		XIPMODE	SCKMODE	Reserved		OPMODE	
Type	RO		RW						RO		RW	RW	RO		RW	
Reset	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0

Table 11-1 QSPI Control Register Description

Field	Name	Description
31:30	Reserved	Reserved
29:24	SCKSCALER	Clock pre-scaler This field defines the scaler factor for generating SCK based on the clock 2 * (value + 1). 00000: F_{sck} = F/2 00001: F_{sck} = F/4 00010: F_{sck} = F/6 00011: F_{sck} = F/8 ... 111111: F_{sck} = F/128 SCK's duty cycle is always 50%.

Field	Name	Description
23:19	Reserved	Reserved
18:16	CSRHT	CSRHT+2 defines the minimum number of CLK cycles in which the chip select (CSn) must remain high between commands issued to the Flash device memory. 000: CSn stays high for at least 2 cycles between Flash memory commands. 001: CSn stays high for at least 3 cycles between Flash memory commands. ... 111: CSn stays high for at least 9 cycles between Flash memory commands.
15:13	Reserved	Reserved
12:8	FMSIZE	Flash memory size This field defines the size of external memory using the following formula: Number of bytes in Flash memory = 2**[FMSIZE + 1] FMSIZE+1 is effectively the number of address bits required to address the Flash memory. The Flash memory capacity can be up to 4GB (addressed using 32 bits), but in the Direct Read mode, it is limited to 256MB since the C-AHB memory space is 512MB. The default value of these fields is 26 (2**27, 128MB). These fields should be set based on the actual memory sizes after power-on by software. If the direct read/write address is overhead in this region, the AHB bus will respond with an error to indicate that the transaction has failed. This field cannot be configured to 5'h1F/5'h0~5'h4. Otherwise, an error will occur.
7:6	Reserved	Reserved
5	XIPMODE	XIP mode select in direct read access mode. This bit indicates that the Flash device memory supports XIP mode by sending alternate bytes. The read instruction only needs to transfer once. Usually, the Flash memory needs to be configured to XIP mode first. 0: The Flash device memory will exit XIP mode 1: The Flash device memory will enter XIP mode.

Field	Name	Description
		This field is valid only in direct read access mode. In XPR mode, this field will be set automatically. When using Indirect mode to enter or exit XIP mode and FSM comes back from Indirect mode, this bit will be set/cleared based on the XIPIMMENTER bit in QSPI_OMCR.
4	SCKMODE	SCK MODE select, Mode 0/Mode 3 This bit indicates the level that SCK takes between instructions when CSn is inactive. 0: SCK must stay low while CSn is inactive. This is referred to as Mode 0. 1: SCK must stay high while CSn is inactive. This is referred to as Mode 3. This field can be modified to different values only when FSM enters an inactive state. It means that the software wants to change this bit. Meanwhile, the opmode bits should be set to 2'b11. Otherwise, this bit cannot be reversed.
3:2	Reserved	Reserved
1:0	OPMODE	Operation mode select 2'b00: Direct read access mode. 2'b01: Reserved. 2'b10: Indirect mode. When an instruction is done, it will come back to the last operation mode automatically (Direct read access mode or inactive state). 2'b11: Inactive mode. It is usually used to assist in configuring the control register. When the current operation mode is indirect access mode, any write to this register will lead to an error response in the APB bus. When writing to 2'b00 or jumping back to 2'b00 from indirect mode, QSPI_RABR/ QSPI_RMCR will be updated to control the new transfer. When writing to 2'b10, QSPI_OMCR/QSPI_OABR/QSPI_IMAR/QSPI_DLR will be used to control the new transfer. When writing to 2'b11, no kick-off registers will be updated.

11.4.2.2 QSPI_SR

0x0004			QSPI status register											QSPI_SR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				FIFODEPTH				Reserved							
Type	RO				RO				RO							
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FIFOLEVEL				Reserved		FIFOFULL	FIFOEMPTY	Reserved	XIPSTATUS	BUSY	TCF	Reserved	OPCRCF	CUR_OP	
Type	RO				RO		RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 11-2 QSPI Status Register Description

Field	Name	Description
31:28	Reserved	Reserved
27:24	FIFODEPTH	This field shows the implementation of FIFO DEPTH. It is just for software to access.
23:16	Reserved	Reserved
15:12	FIFOLEVEL	This field gives the number of valid bytes held in the FIFO. FIFOLEVEL = 0 when the FIFO is empty, and FIFOLEVEL = FIFODEPTH when the FIFO is full. In read mode, there is FIFOLEVEL data in the FIFO that needs to be read back. While, in write mode, it means that there is FIFOLEVEL data in the FIFO that needs to be sent. These bits are used in indirect mode.
11:10	Reserved	Reserved

Field	Name	Description
9	FIFOFULL	This bit is set when the FIFO is full. This bit is usually used in indirect mode.
8	FIFOEMPTY	This bit is set when the FIFO is empty. This bit is usually used in indirect mode. Writing 1 to this field will reset the FIFO (both RX and TX FIFO). To reset the FIFO, you must confirm that your expected transaction has been done. Or you can reset the FIFO before starting a new transaction.
7	Reserved	Reserved
6	XIPSTATUS	This bit indicates that the core is in XIP mode. When the XPR function is enabled and QSPI enters XPR mode, this bit will change to 1 to indicate that QSPI is in XIP mode, and the instruction phase will be discarded. When the XIPIMMENTER bit in QSPI_OMCR is set, and FSM comes back from indirect access mode, this bit will be set to 1. Otherwise, this bit will be cleared to 0. This bit will be set/cleared with TCF set at the same time.
5	BUSY	This bit is set when an operation on QSPI-Bus is ongoing. It means that the SCK wire is toggled in the QSPI bus.
4	TCF	Transfer complete flag This bit is set in indirect mode when the programmed number of data has been transferred. It is cleared by writing 1 to TCF.
3	Reserved	Reserved
2	OPCRCF	Operation mode change request complete flag It indicates that the latest operation mode change request has been granted. When writing the QSPI_CR register, this bit will be cleared to 0 automatically. When the operation mode has been changed, this bit will be set to 1.
1:0	CUR_OP	Current operation mode 2'b00: Direct read mode 2'b01: Reserved

Field	Name	Description
		2'b10: Indirect access mode 2'b11: Inactive mode

11.4.2.3 QSPI_RMCR

0x0008			QSPI direct read access mode control register											QSPI_RMCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DDRMODE	NOWRAP	WRAPMODE	NUMDC					RXDLY		DSIZE		DMODE		ABSIZE	
Type	RW	RW	RW	RW					RW		RW		RW		RW	
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ABMODE		ADSIZE		ADMODE		IMODE		INSTRUCTION							
Type	RW		RW		RW		RW		RW							
Reset	0	0	1	0	0	1	0	1	0	0	0	0	0	0	1	1

Table 11-3 QSPI Direct Read Access Mode Control Register Description

Field	Name	Description
31	DDRMODE	This bit sets the DDR mode for the address, alternate byte, and data phase: 0: DDR mode disabled 1: DDR mode enabled
30	NOWRAP	QSPI does not support WRAP in direct read mode. This bit indicates whether QSPI should support wrap mode. 0: QSPI should follow the AHB transaction, and the critical word reads back first. 1: QSPI does not follow the AHB transaction always. It can be read in an 8-word boundary, and the critical word may not be read back first. This bit can improve read efficiency in some scenarios.
29	WRAPMODE	Flash memory device supports wrap transaction mode selection in direct read access mode.

Field	Name	Description
		This bit indicates that the Flash device memory supports wrap transaction mode. The wrapping bursts read transaction will wrap at 32-byte address boundaries. 0: The Flash device memory is not in wrap mode. 1: The Flash device memory is in wrap mode. This field can be modified only when the operation mode enters an inactive state and BUSY = 0. This field is valid only in direct read access mode. The burst wrap type only supports 8-word cacheable access to suit the cache line length. QSPI_RMCR.NOWRAP and QSPI_RMCR.WRAPMODE cannot be both set to 1'b1, otherwise, it will cause an unpredictable case.
28:24	NUMDC	Number of dummy cycles This field defines the duration of the dummy phase. In both SDR and DDR modes, it specifies a number of SCK cycles (0~31).
23:22	RXDLY	RX delay This field determines the timing for the sampling of the input serial data. It specifies the number of delay steps for capturing the serial data inputs. Each delay step is equal to one core bus clock cycle. If it is set to 0, the input serial data is captured with the rising edge of the generated SPI clock. Supported values range from 0 to 3. 00: 0 AHB bus clock delay 01: 1 AHB bus clock delay 10: 2 AHB bus clock delay 11: 3 AHB bus clock delay
21:20	DSIZE	Data size This field defines the data size: 00: Reserved

Field	Name	Description
		01: Reserved 10: Reserved 11: 32 bits of data It only supports 32 bits of data. This field is always 2'b11 and read-only.
19:18	DMODE	Data mode This field defines the data phase's mode of operation: 00: Reserved 01: Data on a single line 10: Data on two lines 11: Data on quad lines
17:16	ABSIZE	Alternate bytes size This field defines the address size: 00: 8-bit alternate bytes. 01: 16-bit alternate bytes. 10: 24-bit alternate bytes. 11: 32-bit alternate bytes.
15:14	ABMODE	Alternate bytes mode This field defines the address phase's mode of operation: 00: No alternate bytes 01: Alternate bytes on a single line 10: Alternate bytes on two lines 11: Alternate bytes on quad lines
13:12	ADSIZE	Address size This field defines the address size: 00: Reserved 01: Reserved 10: 24-bit address

Field	Name	Description
		11: 32-bit address
11:10	ADMODE	Address mode This field defines the address phase's mode of operation: 00: No address bytes 01: Address on a single line 10: Address on two lines 11: Address on quad lines
9:8	IMODE	Instruction mode This field defines the instruction phase mode of operation: 00: No instruction bytes 01: Instruction on a single line 10: Instruction on two lines 11: Instruction on quad lines
7:0	INSTRUCTION	Direct read access mode instruction sent to the SPI device. It is the arbitrary instruction to be sent to the external Flash device memory in any access mode.

11.4.2.4 QSPI_OMCR

0x000c			QSPI indirect mode control register											QSPI_OMCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DDRMODE	IDMODE	XIPIMMENTER	NUMDC					RXDLY		DSIZE		DMODE		ABSIZE	
Type	RW	RW	RW	RW					RW		RW		RW		RW	
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ABMODE		ADSIZE		ADMODE		IMODE		INSTRUCTION							
Type	RW		RW		RW		RW		RW							
Reset	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0

Table 11-4 QSPI Indirect Mode Control Register Description

Field	Name	Description
31	DDRMODE	This bit sets the DDR mode for the address, alternate byte and data phase: 0: DDR mode disabled. 1: DDR mode enabled.
30	IDMODE	Indirect access mode data phase direction. 0: Read data, RX FIFO enabled. 1: Write data, TX FIFO enabled. This bit is valued only when OPMODE_SEL = 2'b10. There is no effect in other operation modes. This field also enables RX/TX FIFO. When writing this bit to change the direction of the FIFO, the FIFO will be reset. When there is data in the RX FIFO, the APB bus should read empty RX FIFO. Then the software can set the bit to 1 to enable the TX FIFO and indirect

Field	Name	Description
		TX mode. When there is data in the TX FIFO, QSPI should read empty TX FIFO. Then the software can set the bit to 0 to enable the RX FIFO and indirect RX mode. Note: When the current indirect access does not finish, this bit should not be changed. If this bit changes, the FIFO's direction will be changed, and an error will be generated. It can only change in non-indirect mode.
29	XIPIMMENTER	This bit indicates that when QSPI's FSM comes back from indirect access, the XIPSTATUS bit in QSPI_SR will be set/cleared to indicate whether the Flash device is in XIP mode. 1'b1 means that it is set, and 1'b0 means that it is cleared.
28:24	NUMDC	Number of dummy cycles. This field defines the duration of the dummy phase. In both SDR and DDR modes, it specifies a number of SCK cycles (0~31).
23:22	RXDLY	RX delay. This field determines the timing for the sampling of the input serial data. It specifies the number of delay steps for capturing the serial data inputs. Each delay step is equal to one core bus clock cycle. If it is set to 0, the input serial data is captured with the rising edge of the generated SPI clock. Supported values range from 0 to 3. 00: 0 AHB bus clock delay. 01: 1 AHB bus clock delay. 10: 2 AHB bus clock delay. 11: 3 AHB bus clock delay.
21:20	DSIZE	Data size This field defines the data size: 00: 8 bits data 01: 16 bits data 10: Reserved 11: 32 bits of data

Field	Name	Description
19:18	DMODE	Data mode This field defines the data phase's mode of operation: 00: Reserved. 01: Data on a single line. 10: Data on two lines. 11: Data on quad lines.
17:16	ABSIZE	Alternate bytes size This field defines the address size: 00: 8-bit alternate bytes. 01: 16-bit alternate bytes. 10: 24-bit alternate bytes. 11: 32-bit alternate bytes.
15:14	ABMODE	Alternate bytes mode This field defines the address phase's mode of operation: 00: No alternate bytes. 01: Alternate bytes on a single line. 10: Alternate bytes on two lines. 11: Alternate bytes on quad lines.
13:12	ADSIZE	Address size This field defines the address size: 00: 8-bit address. 01: 16-bit address. 10: 24-bit address. 11: 32-bit address. Indirect mode supports all address sizes.
11:10	ADMODE	Address mode This field defines the address phase's mode of operation:

Field	Name	Description
		00: No address bytes. 01: Address on a single line. 10: Address on two lines. 11: Address on quad lines.
9:8	IMODE	Instruction mode This field defines the instruction phase mode of operation: 00: No instruction bytes. 01: Instruction on a single line. 10: Instruction on two lines. 11: Instruction on quad lines.
7:0	INSTRUCTION	Indirect access mode instruction sent to the SPI device. It is the arbitrary instruction to be sent to the external Flash device memory in any access mode.

11.4.2.5 QSPI_RABR

0x0010			QSPI direct read mode alternate bytes register											QSPI_RABR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	ALT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ALT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-5 QSPI Direct Read Mode Alternate Bytes Register Description

Field	Name	Description
31:0	ALT	Alternate bytes. Optional data to be sent to the external SPI device after the address phase. QSPI will send LSB 0/8/16/24/32 bits to Flash according to ABMODE and ABSIZE in QSPI_DMCR. This field can be used in direct read access mode.

11.4.2.6 QSPI_OABR

0x0014			QSPI indirect mode alternate bytes register											QSPI_OABR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	ALT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ALT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-6 QSPI Indirect Mode Alternate Bytes Register Description

Field	Name	Description
31:0	ALT	Alternate bytes. Optional data to be sent to the external SPI device after the address phase. QSPI will send LSB 0/8/16/24/32 bits to Flash according to ABMODE and ABSIZE in QSPI_DMCR. This field can be used in indirect access mode.

11.4.2.7 QSPI_IMAR

0x0018			QSPI indirect mode address register											QSPI_IMAR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	IMADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IMADDR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-7 QSPI Indirect Mode Address Register Description

Field	Name	Description
31:0	IMADDR	Address to be sent to the external SPI device. This field is only used in indirect access mode.

11.4.2.8 QSPI_FDR

0x001c			QSPI FIFO data register											QSPI_FDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	FIFODATA															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FIFODATA															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-8 QSPI FIFO Data Register Description

Field	Name	Description
31:0	FIFODATA	Data to be sent/received to/from the external SPI device. This field is only used in indirect access mode. 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 full, a write operation will respond with an APB error. In indirect read mode, reading this register gives the data which is received from the Flash memory. If the FIFO is empty, a read operation will respond with an APB error. The FIFO will hold the APB bus transaction to hide the hardware operation. This register is for word accesses only. But RX/TX data can be organized by word, half-word, and byte. It is determined by the DSIZE bit in QSPI_OMCR. For example, 3 bytes need APB to read 3 times in byte mode. 4 bytes need APB to read 2 times in halfword mode.

Field	Name	Description
		NOTE: It is strongly recommended that if the software wants to operate the FIFO, it must first check the FIFO status.

11.4.2.9 QSPI_DLR

0x0020			QSPI indirect mode data length register											QSPI_DLR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved												DL			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-9 QSPI Indirect Mode Data Length Register Description

Field	Name	Description
31:20	Reserved	Reserved
19:0	DL	Transfer data length. Number of data to be retrieved (DL + 1) in indirect read/write mode. 0x00000: 1 byte is to be transferred. 0x00001: 2 bytes are to be transferred. 0x00002: 3 bytes are to be transferred. ... 0xFFFFF: 1,048,576(1M) bytes are to be transferred. This field has no effect when in direct read mode. **NOTE:** Software can guarantee the consistency between the transmit data length and the DSIZE bit in QSPI_OMCR.

Field	Name	Description
		If DSIZE uses 16 bits, DL[0] must be 1. If DSIZE uses 32 bits, DL[1:0] must be 2'b11. Otherwise, hardware will force to be consistent between the two registers. It means that DSIZE uses 16 bits, DL[19:1] will be loaded to counter, and DL[0] is always 1'b1. DSIZE uses 32 bits, DL[19:2] will be loaded to counter, and DL[1:0] is always 2'b11.

11.4.2.10 QSPI_WCNT

0x0024			QSPI wait counter register											QSPI_WCNT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		WCNT													
Type	RO		RW													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	WCNT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-10 QSPI Wait Counter Register Description

Field	Name	Description
31:30	Reserved	Reserved
29:0	WCNT	Wait counter value When programming Flash memory non-volatile registers/arrays while executing routines from Flash, if these bits are set (WCNT != 0), and after QSPI configures the non-volatile registers/memory arrays, FSM will not come back to direct read mode from indirect mode until WCNT cycles have waited. Any direct read access from AHB will be pending if the timer is not reached.

Analog-to-Digital Converter (ADC12)

This chapter describes the details of the Analog-to-Digital Converter (ADC12).

Topics: **Page**

12.1 Introduction

The Analog-to-Digital Converter (ADC) is a high-performance 8-bit, 10-bit, or 12-bit ADC. It offers support for single-end mode or differential mode, hardware oversampling, analog watchdog, and self-calibration for gain and offset.

12.2 Features

- Up to 20-ch supported, see device specific for details.
- Differential or single-end support
- Monotonic 8-bit, 10-bit or 12-bit converter with no missing codes
- AHB clock or adc_pclk clock source configurable for ADC clock
- ADC conversion time can be independent from the AHB bus clock frequency
- Hardware oversampling ratio adjustable from 2 to 256
- Auto calibration with gain and offset compensation
- Analog watchdog
- ADC input range: $V_{REF-} \le V_{IN} \le V_{REF+}$

12.3 Functional Description

The operation of the ADC is described in the following sections.

12.3.1 Block Diagram

Figure 12-1 shows the block diagram of the ADC module.

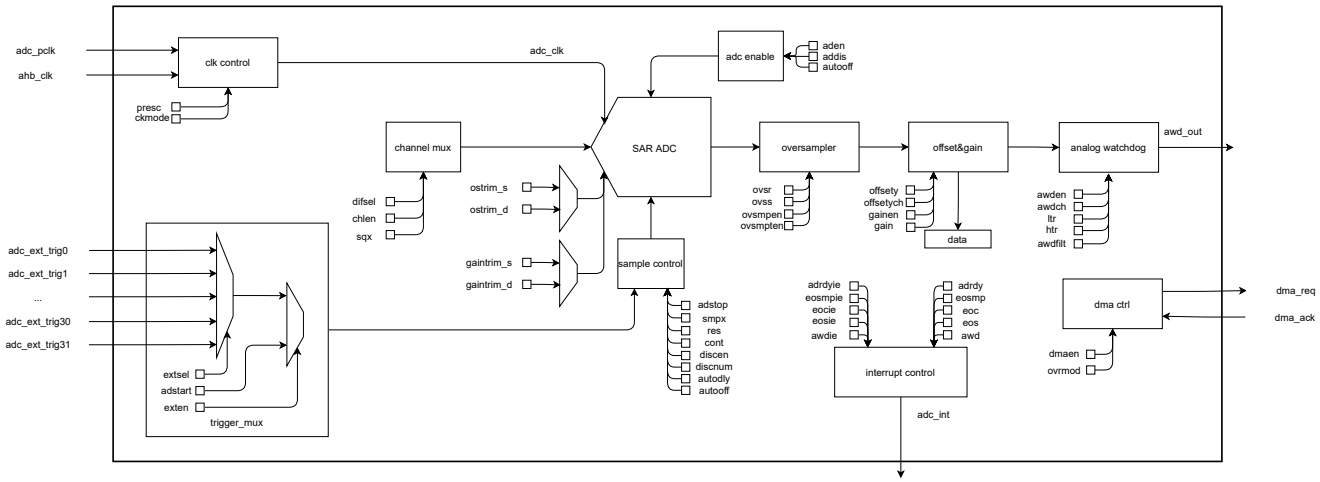


Figure 12-1 ADC Block Diagram

12.3.2 ADC Clock

The ADC clock can be a specific clock source derived from the following clock sources:

- SYSCLK
- PCLK (adc_pclk))

See Reset and Clock Control (RCC) for details. Figure 12-2 shows the block diagram of ADC Clock.

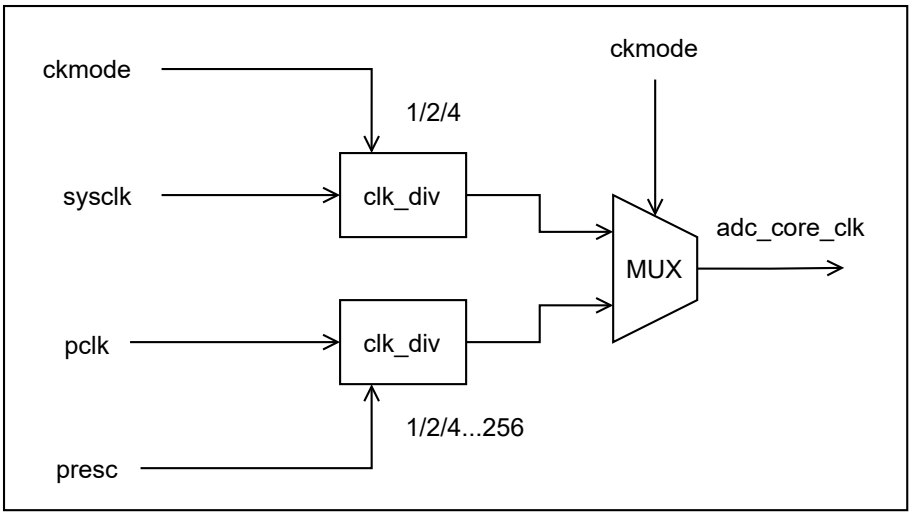


Figure 12-2 ADC Clock Block Diagram

ADC clock can be derived from the HCLK (AHB bus) of the ADC bus interface, divided by 1, 2, 4. See the following table for details:

Item	Details			
CKMODE	0	1	2	3
Clock source	PCLK (adc_pclk)	HCLK	HCLK/2	HCLK/4

ADC clock has pre-divider for adc_pclk, see the following table for details:

Item	Details											
PRESC	0	1	2	3	4	5	6	7	8	9	a	b
Divided	1	2	4	6	8	10	12	16	32	64	128	256

Slave AHB Interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests and never generates AHB errors.

12.3.3 ADC Input Channels

The ADC module supports up to 20 input channels. The channels can be configured to be either single-ended input or differential input by programming DIFSEL bits in the ADC_DIFSEL register. The SQRx are selected for these channels. Refer to the device specification for details.

Table 12-1 ADC Input Channel

Channel Number	Channel Type	Description
0~13	External	External input channel: ADC_IN0~ADC_IN13

Channel Number	Channel Type	Description
14	Internal	Internal OA0_OUT
15	Internal	Internal OA1_OUT
16	Internal	Internal TS
17	Internal	Internal 1/2 VBAT input
18	Internal	Internal 1/2 VDDA input
19	Internal	Internal VREFBUF

12.3.4 ADC Differential Mode

The ADC supports either single mode (DIFSEL = 0) or differential mode. To enable differential mode, set DIFSEL bits according to the application requirements. Figure 12-3 shows the differential mode block diagram.

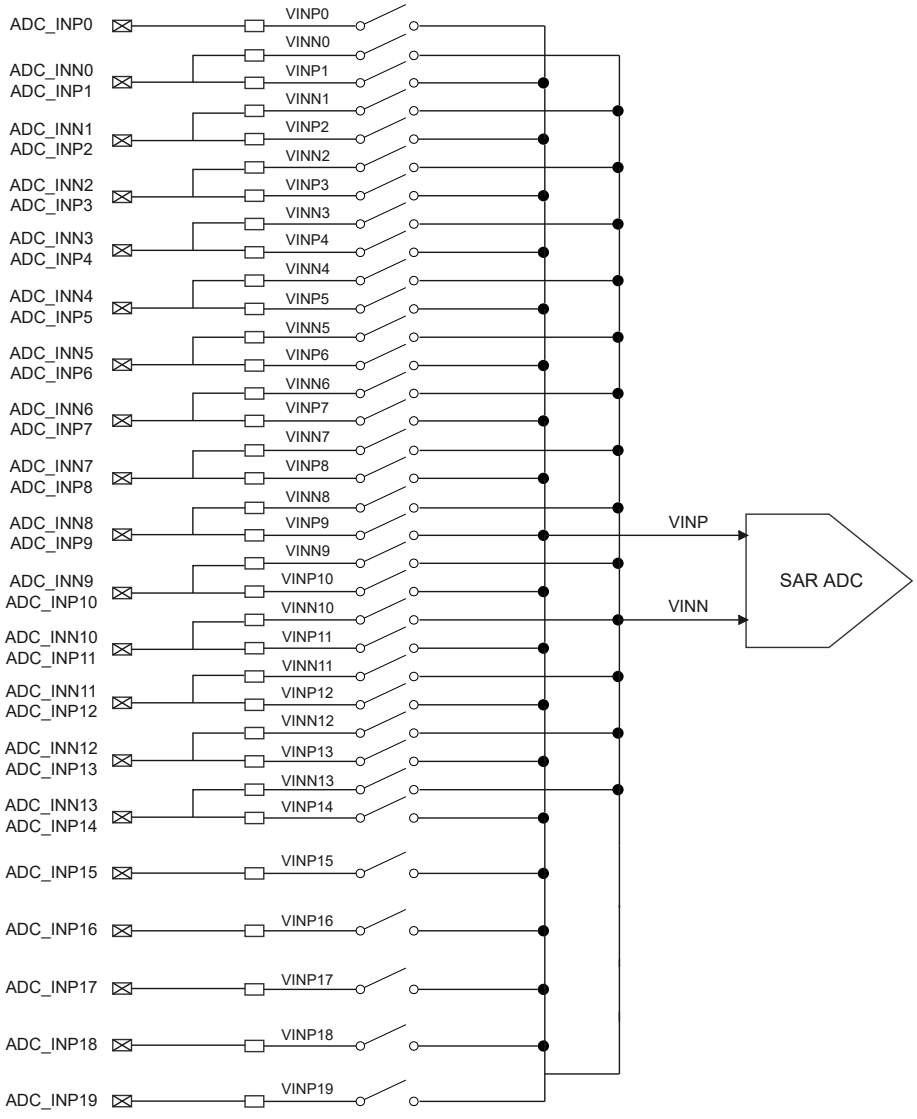


Figure 12-3 ADC Differential Mode

12.3.5 ADC Core

The ADC core converts an analog input to its 8-bit, 10-bit, or 12-bit digital representation and stores the result in the conversion register ADC_DR.

The core uses two programmable and selectable voltage levels (V_{REF+} and V_{REF-}) to define the upper and lower limits of the conversion. The digital output (N_{ADC}) is full-scale (03FFh for 10-bit and 0FFFh for 12-bit) when the input signal is equal to or higher than V_{REF+}. The output is zero when the input signal is equal to or lower than V_{REF-}. The input channel and the reference voltage levels (V_{REF+} and V_{REF-}) are defined in the conversion control register. The conversion formula for the ADC result N_{ADC} is:

$$\text{DATA (ADC result)} = 4096 \times V_{adc\_in} / (V_{REF+} - V_{REF-}) \quad (1)$$

12.3.6 ADC Trigger

The ADC module supports either software or external hardware trigger events.

- Software trigger is effective once ADSTART = 1.
- External hardware trigger supports rising edge, falling edge, and both edges. See the following table:

EXTEN	Description
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges (interval these edges shall be longer than 4x ADC_CLK)

The polarity of the regular trigger cannot be changed on-the-fly. Refer to device specification for trigger source details.

12.3.7 ADC On & Off Control

Two control bits enable or disable the ADC:

- ADCEN = 1 enables the ADC. The flag ADRDY will be set once the ADC is ready for operation.
- ADDIS = 1 disables the ADC. ADCEN and ADDIS are then automatically cleared by hardware as soon as the ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART = 1 or when an external trigger event occurs if triggers are enabled.

Figure 12-4 shows the block diagram of enabling/disabling ADC flow.

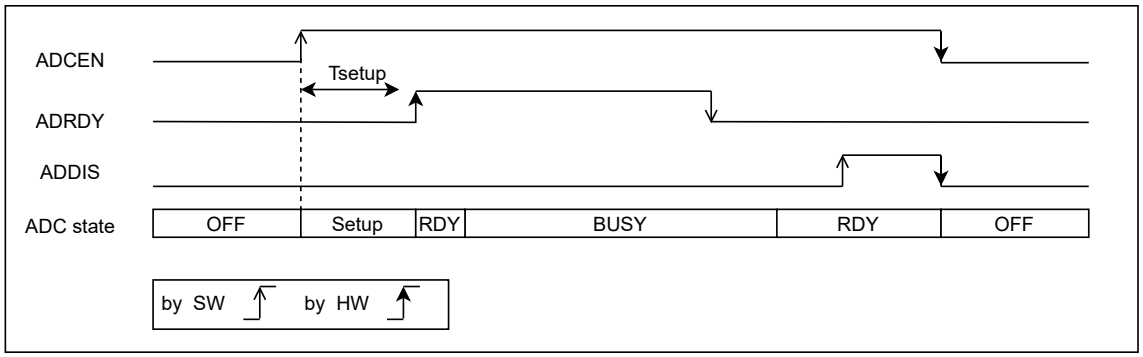


Figure 12-4 Enabling/Disabling ADC Flow

12.3.7.1 Enabling ADC

To enable the ADC flow, follow these steps:

1. Clear the ADRDY bit in the ADC_ISR register by writing “1”.
2. Set ADCEN = 1.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be accomplished using the associated interrupt (by setting ADRDYIE = 1).
4. Optionally, the ADRDY bit in the ADC_ISR register can be reset by writing “1”.

NOTE: The ADCEN bit cannot be set when the OSADCALI or GAINADCALI bits are set. Additionally, once the OSADCALI or GAINADCALI bit is cleared by hardware (indicating the completion of the calibration process), the ADCEN bit cannot be set for the subsequent 4xADC clock cycles.

12.3.7.2 Disabling ADC

To disable the ADC flow, follow these steps:

1. Check that ADSTART = 0 to ensure that no conversion is ongoing. If required, stop any regular ongoing conversion by setting ADSTOP = 1 and wait until ADSTOP = 0.
2. Set ADDIS = 1.
3. If required by the application, wait until ADCEN = 0, which indicates that the ADC is effectively disabled. Once ADCEN = 0, ADDIS will automatically reset.

12.3.8 Calibration

ADC provides an automatic calibration procedure that drives all the calibration sequences, including the power-on/off sequence of the ADC. During this procedure, the ADC calculates a 5-bit calibration factor that is internally applied to the ADC until power-off.

NOTE: The ADCEN bit cannot be set under the following conditions:

- When either the OSADCALI or the GAINADCALI bit is set.
- During the 4 x ADC clock cycles that follow the clearing of the OSADCALI or the GAINADCALI bit by hardware (that is, at the end of calibration).

Calibration is preliminary to any ADC operation. It removes the offset/gain error, which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALIDIF = 0 before launching a calibration, which will be applied for single-ended input conversions.
- Write ADCALIDIF = 1 before launching a calibration, which will be applied for differential input conversion.

OSADCALI or GAINADCALI = 1 is initialized by software for calibration, and it is only enabled when ADC is disabled (ADCEN = 0). OSADCALI or GAINADCALI bit stays at 1 during all calibration sequences. It is then automatically cleared when the calibration is complete. At this time, the associated calibration factor is stored in the ADC_CALFACT register.

The calibration is kept if the ADC is disabled (ADCEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

The calibration factor can be written if the ADC is enabled but not converting (ADCEN = 1 and ADSTART = 0). Then, at the next start of conversion, the calibration factor will automatically be injected into the ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. Recalibration is recommended when V_{REF+} voltage changes more than 10%.

12.3.9 Programmable Sampling Timer

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level. Each channel can be sampled with a different sampling time, which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2, SMPPLUS in the ADC_CFGR2.

12.3.10 Conversion Modes

12.3.10.1 Single Conversion Mode

In single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel)
- External hardware trigger event (for a regular channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (End of Conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the regular sequence is complete:

- The EOS (End of Sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then the ADC stops until a new external regular occurs or until the ADSTART bit is set again.

NOTE: To convert a single channel, program a sequence with a length of 1.

12.3.10.2 Continuous Conversion Mode

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversion of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (End of Conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (End of Sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately, and the ADC continuously repeats the conversion sequence.

12.3.10.3 Discontinuous Conversion Mode

Regular Group Mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register to 1. It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register, n = DISCNUM + 1.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQy registers until all the conversions in the sequence are done. The total sequence length is defined by CHLENGTH bits.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 4, 7, 8, 9, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion)
  - 2nd trigger: channels converted are 4, 7, 8 (an EOC event is generated at each conversion)
  - 3 rd trigger: channels converted are 9, 11 (an EOC event is generated at each conversion and the last one generates an EOS event)
  - 4th trigger: channels converted 1, 2, 3 (an EOC event is generated at each conversion)
- DISCEN = 0, channels to be converted = 1, 2, 3, 4, 7, 8, 9, 11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 4, 7, 8, 9 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events will relaunch the complete sequence.

12.3.10.4 Starting Conversion

Software starts ADC conversion by setting ADSTART = 1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 00 (software trigger)
- At the next active edge of the selected regular hardware trigger

ADSTART is cleared by hardwares.

- In single mode with software trigger (CONT = 0, EXTSEL = 0x0)

  At any end of conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT = x, EXTSEL = x)

  After execution of the ADSTOP procedure asserted by the software.

12.3.11 ADC Data Management

Data Alignment

At the end of each regular conversion channel (when an EOC event occurs), the result of the converted data is stored in the ADC_DR data register, which is 16-bit wide. The ALIGN bit in the ADC_CFGR register selects the alignment of the data stored after conversion. The data can be right- or left-aligned, as shown in Figure 12-5 to Figure 12-8.

Offset

An offset y (y = 1, 2, 3, 4) can be applied to a channel by setting the bit OFFSETyEN = 1 into the ADC_OFRy register. The channel to which the offset will be applied is programmed into the bits OFFSETyCH[4:0] of the ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSETy[11:0]. The result may be a negative value, so the read data is signed, and the SEXT bit represents the extended sign value.

NOTE: Offset correction is not supported in oversampling mode. When the OVSMPEN bit is set, the value of the OFFSETyEN bit in the ADC_OFRy register is ignored (considered as reset).

Table 12-2 Offset Computation vs. Data Resolution

Resolution	RES	Subtraction between Raw Converted Data and Offset		Result	Comments
		Raw Converted Data, Left Aligned	Offset		
12-bit	00	DATA [11:0]	OFFSET [11:0]	Signed 12-bit data	-
10-bit	01	DATA[11:2], 00		Signed 10-bit data	Must configure OFFSET[1:0] to 00
8-bit	10	DATA[11:4], 0000		Signed 8-bit data	Must configure OFFSET[3:0] to 0000
Reserved	11	Reserved			

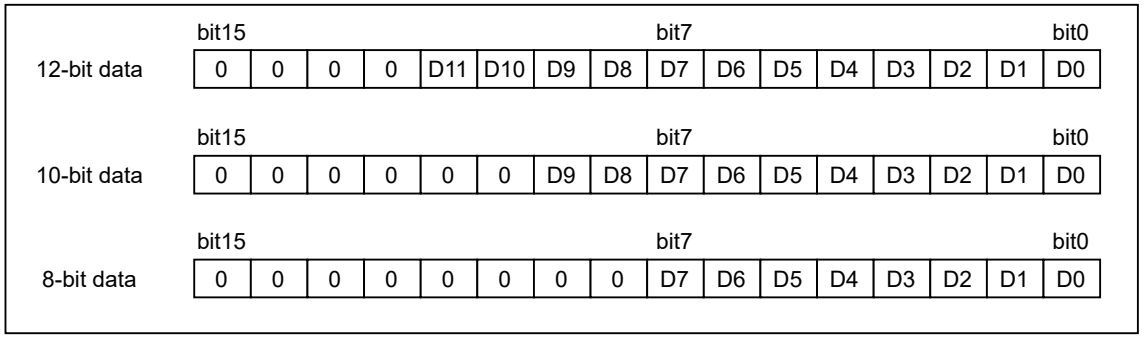


Figure 12-5 Right Alignment (Offset Disabled, Unsigned Value)

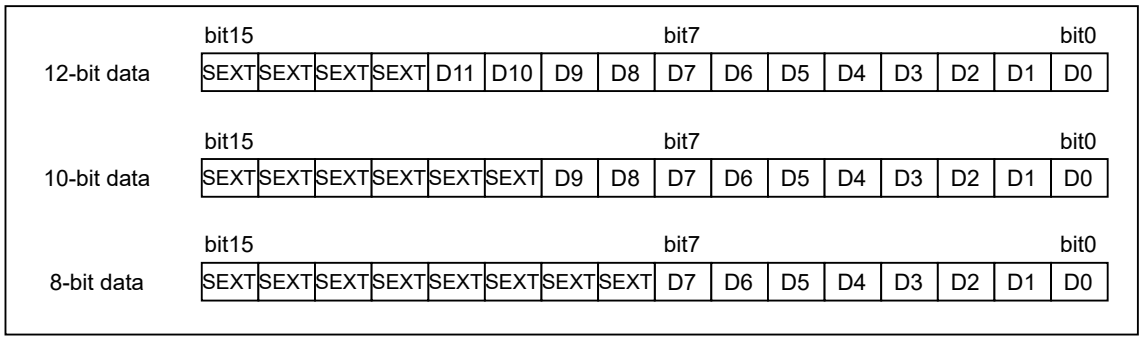


Figure 12-6 Right Alignment (Offset Enabled, Signed Value)

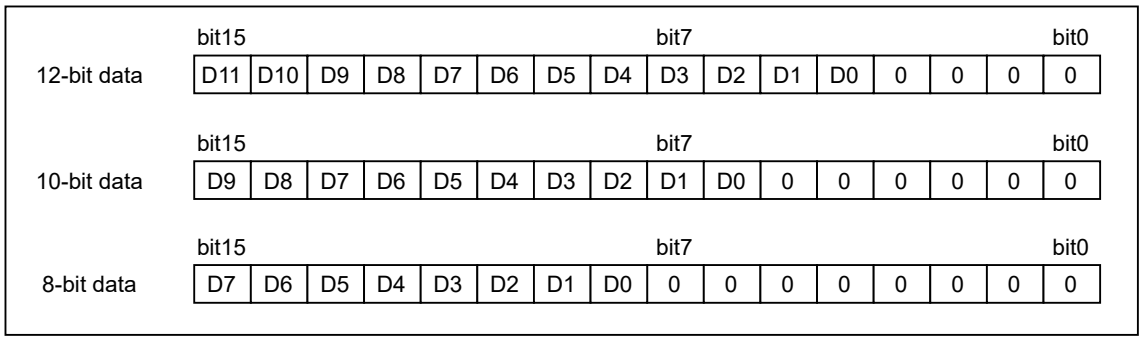


Figure 12-7 Left Alignment (Offset Disabled, Unsigned Value)

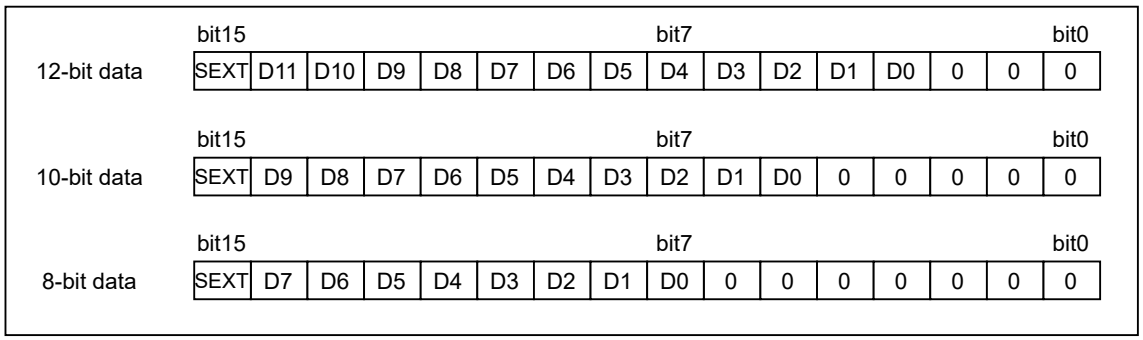


Figure 12-8 Left Alignment (Offset Enabled, Signed Value)

Gain Compensation

The gain compensation is activated on all the converted data when GAINEN = 1. After each conversion, data is calculated with Equation 2.

$$DATA = (ADC\ result + OFFSET) \times GAINCOEF / 16384 \quad (2)$$

As GAINCOEF can be programmed from 0 to 16383, the actual gain compensation factor can range from 0 to 0.999939.

Before storing the resulting data in RDATA registers, the LSB−1 value is evaluated to round up the data and minimize the error.

The gain compensation is also effective for oversampling. When the gain compensation is used for the oversampling mode, the gain calculation is performed after the accumulation and right-shift operations to minimize the power consumption (the gain calculation is done only once instead of at each conversion).

Offset Compensation

Enabling the SATEN bit in the ADC_OFRy register during offset operations sets the data format to unsigned. All the offset data saturate at 0x000 (in 12-bit mode). When the OFFSETPOS bit is set, the offset direction is positive, and the data saturate at 0xFFF (in 12-bit mode). In 8-bit mode, data saturate at 0x00 and 0xFF, respectively.

The analog watchdog comparison is performed on unsigned values after offset and gain compensation. For correct comparison operation, the data after offset compensation must be in the unsigned format, requiring the SATEN bit be set to 1 in ADC_OFRy.

ADC Overrun

The overrun flag (OVR) notifies that a buffer overrun event occurred when the regular converted data had not been read (by the CPU or the DMA) before newly converted data became available.

The OVR flag is set if the EOC flag is still 1 when a new conversion completes. An interrupt can be generated if bit OVRIE = 1. When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting bit ADSTOP = 1 . The software clears the OVR flag by writing 1 to it.

OVRMOD bit sets the data preserved or overwritten when an overrun event occurs:

- OVRMOD = 0: The overrun event preserves the data register from being overrun. The old data is maintained, and the new conversion is discarded and lost. If OVR remains at 1, any further conversions will occur, but the resulting data will also be discarded.
- OVRMOD = 1: The data register is overwritten with the last conversion result, and the previous unread data is lost. If OVR remains at 1, any further conversions will operate normally, and the ADC_DR register will always contain the latest converted data.

ADC DMA

ADC supports DMA requests for data transfer from the dedicated register to memory.

- Managing a sequence of conversions without using the DMA

  If the conversions are slow enough, the conversion sequence can be handled by the software. In this case, the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD should be configured to 0 to manage overrun events as an error. DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.
- Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog, for instance). In this case, the OVRMOD bit must be configured to 1 and the OVR flag should be ignored by the software. An overrun event will not prevent the ADC from continuing to convert, and the ADC_DR register will always contain the latest conversion.

- Managing conversions using the DMA

  Since converted channel values are stored in a unique data register, it is useful to use DMA for the conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

  When the DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR register) in single ADC mode, a DMA request is generated after each channel conversion. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

  Depending on the configuration of the OVRMOD bit, the data is either preserved or overwritten.

12.3.12 ADC Timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution, as shown in Equation 3.

$$t_{CONV} = t_{SMPL} + t_{SAR} \quad (3)$$

Figure 12-9 shows the block diagram of analog to digital conversion time.

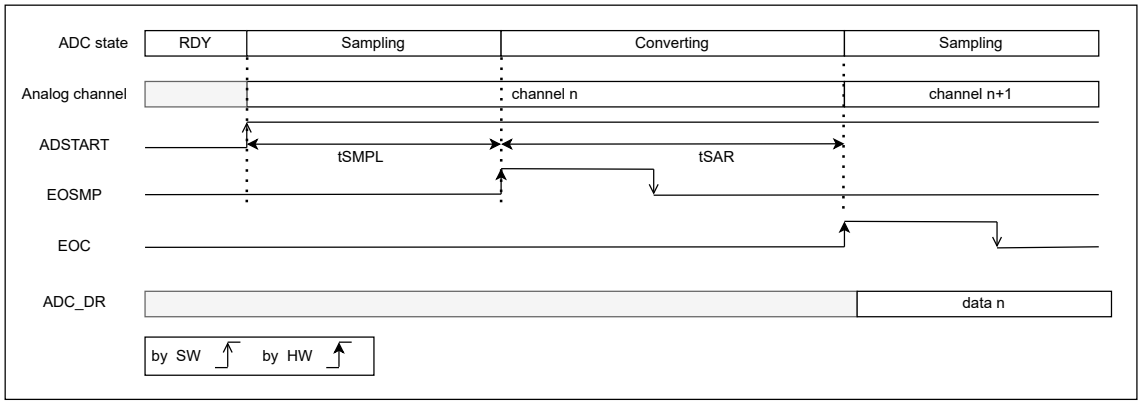


Figure 12-9 Analog to Digital Conversion Time

(1) t_{SMPL} depends on SMP[2:0].

(2) t_{SAR} depends on RES[2:0].

12.3.13 ADC Stopping

The software can be set to stop conversions ongoing by ADSTP = 1. Stopping conversions will reset the ongoing ADC conversions operation, and then the ADC can be reconfigured to be ready for the new operations.

When software sets the ADSTP bit, any ongoing regular conversion is aborted with partial result discarded.

NOTE: ADC_DR register is not updated with the current conversion.

Once this procedure is completed, ADSTP/ADSTART bits are cleared by hardware, and the software must poll ADSTART until the bit is reset before assuming the ADC is completely stopped. Figure 12-10 shows the block diagram of stopping ongoing regular conversions.

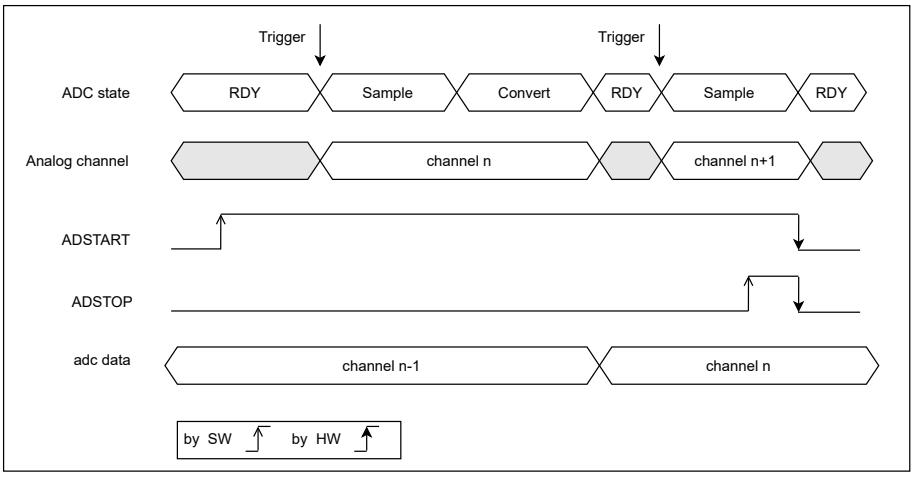


Figure 12-10 Stopping Ongoing Regular Conversions

12.3.14 ADC Dynamic Low-Power

12.3.14.1 Auto Delay

The ADC implements an auto-delayed conversion mode controlled by the AUTODLY bit. Auto-delayed conversions are useful for simplifying software and optimizing the performance of an application clocked at a low frequency, where there is a risk of encountering an ADC overrun.

When AUTODLY = 1, a new conversion can only start under the following conditions:

- All previous data has been treated.
- The ADC_DR register has been read.
- The EOC bit has been cleared.

This feature allows the ADC to automatically adjust its speed to match the speed of the system that reads the data.

NOTE:
- A delay is inserted after each regular conversion, regardless of whether DISCEN = 0 or 1.
- During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

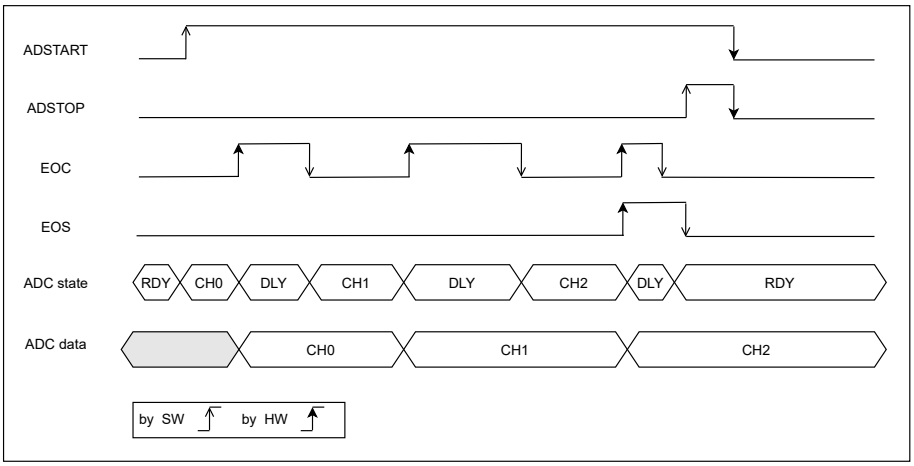


Figure 12-11 AUTODLY = 1, Regular Conversion in Continuous Mode, Software Trigger

12.3.14.2 Auto Off

The ADC implements an auto-off mode that is controlled by the AUTOFF bit. This mode automatically disables the ADC after a group conversion is completed (EOS = 1), and then enables the ADC when new trigger events are asserted. However, it is important to allow sufficient time for the ADC to settle.

NOTE: The auto-off mode is not supported on continuous mode.

12.3.15 ADC Oversampling

The oversampling unit performs data pre-processing to offload the CPU. It is capable of handling multiple conversions and averaging them into a single data with an increased data width of up to 16 bits. The result is provided by Equation 4, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n) \quad (4)$$

It allows performing by hardware the following functions:

- Averaging
- Data rate reduction
- Signal-to-Noise Ratio (SNR) improvement
- Basic filtering

The oversampling ratio N is defined using the OVSR bits in the ADC_CFGR2 register, and can range from 2x to 256x. The division coefficient M comprises a right bit shift up to 8 bits and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result of up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to the 16 least significant bits, rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

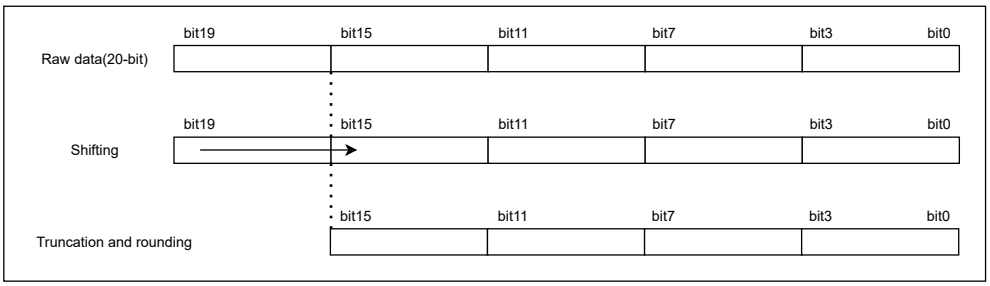


Figure 12-12 20-bit to 16-bit Result Truncation

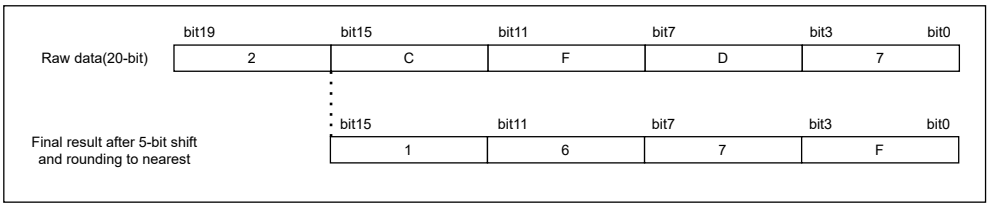


Figure 12-13 Numerical Example with 5-bit Shift and Rounding

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. New data is provided for every N conversion, with an equivalent delay equal to $N \times T_{CONV} = N \times (T_{SMPL} + T_{SAR})$. The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase.
- The end of conversion (EOC) occurs once every N conversion, when the oversampled result is available.
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is, after N x sequence length conversions total).

ADC operating modes supported when oversampling (single ADC mode)

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous mode conversions
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTODLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR1 register) are accumulated, truncated, rounded and shifted in the same manner as 12-bit conversions.

12.3.16 ADC DMA

ADC supports DMA request since converted channel values are stored into a unique data register.

12.3.17 ADC Analog Watchdog

The analog watchdog allows the ADC to monitor analog signals without any CPU interaction. The analog watchdog is enabled by setting the AWDCH field in the ADC_AWDCR register. It monitors all the enabled channels that remain within a configured voltage range (window). Figure 12-14 shows the block diagram of the ADC analog watchdog.

Figure 12-14 ADC Analog Watchdog Block Diagram

An interrupt can be enabled by setting AWDIE in the ADC_IER register. ADW flag is cleared by software with writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment. To program the analog watchdog, the Higher Threshold (HTH) and Lower Threshold (LTH) values in the ADC_AWDTR register must be set correctly. It is important to ensure that the values in the registers are in the correct data format. When converting data with a resolution of less than 12 bits (according to bits RES[1:0]), the LSB of the programmed thresholds should be kept clear. This is because the internal comparison is always performed on the full 12-bit raw converted data, which is left-aligned.

Resolution	RES	DATA	Threshold	Comments
12-bit	00	DATA [11:0]	HTH[11:0] and LTH[11:0]	-
10-bit	01	DATA[11:2], 00		Must configure HTH [1:0] and LTH[1:0] to 00
8-bit	10	DATA[11:4], 0000		Must configure HTH [3:0] and LTH[3:0] to 0000
Reserved	11	Reserved	Reserved	Reserved

Analog Watchdog DMA EN

When an ADC is configured with a single input channel (no scan mode) and the DMAEN in the ADC_CFGR is set:

- If the AWDDMAEN = 1 in the ADC_AWDTR register, the ADC conversion data in the range of the AWD window will generate a DMA request.
- If the AWDDMAEN = 0 in the ADC_AWDTR register, all the ADC conversion data will generate a DMA request.

Analog Watchdog Filter

When an ADC is configured with a single input channel (no scan mode), you can configure a valid ADC conversion data interval through the ADC_AWDTR register.

- If the ADC conversion data falls outside the specified range fewer times than the value set in the AWDFILT bit of ADC_AWDTR, no AWD flag is generated.
- However, if the data falls outside the specified range more times than the value set in the AWDFILT bit of ADC_AWDTR, the AWD flag is set, and the corresponding interrupt is triggered.

Analog Watchdog Out

An analog watchdog is associated with an internal hardware signal ADC_AWD_OUT, which is directly connected to some on-chip timers' ETR input (external trigger). Refer to the timers section for details.

- ADC_AWD_OUT is active when the associated analog watchdog is enabled.
- ADC_AWD_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWD_OUT is reset after the end of the next guarded conversion that falls within the programmed thresholds.

  However, if subsequent guarded conversions still fall outside the thresholds, ADC_AWD_OUT remains set at 1.
- Additionally, ADC_AWD_OUT is reset when the ADC is disabled by setting ADDIS = 1.

NOTE: Stopping the conversions does not affect the generation of ADC_AWD_OUT.

12.3.18 ADC Temperature Sensor

The on-chip temperature sensor can be used to measure the Junction Temperature (Tj) of the microcontroller. The temperature sensor is internally connected to the Analog-to-Digital Converter (ADC) input channels, which convert the sensor's output voltage to a digital value for further

processing. The sensor can be put in power-down mode when not in use to conserve energy. The temperature sensor supports a temperature range of -40 to 125 °C.

During the manufacturing process, the calibration data of the temperature sensor is stored in the system memory area (TS_125 & TS_25). The user application can read this data and utilize it to improve the accuracy of the temperature sensor. By applying the calibration data, the temperature sensor can achieve ±1°C accuracy within the temperature range of -40 to 125 °C.

12.3.19 ADC Internal Voltage Monitoring

ADC can monitor V_{BAT}, V_{DDA} and V_{REF} voltages. Refer to device-specific for a detailed input channel. Figure 12-15 shows the block diagram of V_{DDA} and V_{BAT} sensing feature.

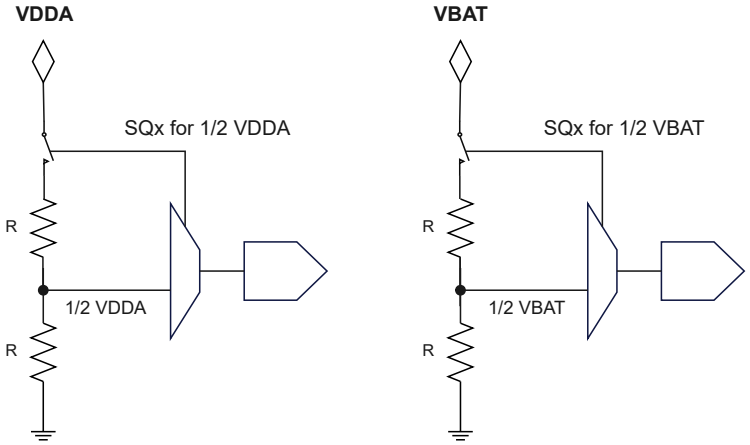


Figure 12-15 V_{DDA} and V_{BAT} Sensing Block Diagram

VREFBUF is directly connected to the ADC input channel, and it can be used to monitor VREFBUF reference voltage and calculate actual V_{REF+} or V_{DDA} voltage.

12.3.20 ADC Boost Mode

When the ADC power supply voltage is between 1.6 V and 2.0 V:

- If the ADC clock is set to low speed (24 MHz), it is recommended to enable the boost function (BOOSTEN = 1) to achieve full performance at the normal voltage level.
- If the ADC clock is set to high speed (45 MHz), even if BOOSTEN is set to 1, the ADC can only operate on function and performance is not guaranteed.

When the ADC power supply voltage is higher than 2.0 V:

BOOSTEN = 0, the ADC performance remains normal for low-speed and high-speed clocks.

12.4 Interrupts

Interrupts can be generated using the following methods:

- ADRDY: ADC ready flag (set after ADC power-up, when the ADC is ready)
- EOC: End of conversion flag (set at the end of any conversion for regular channels)
- EOS: End of sequence of conversion flag (set at the end of a sequence of conversion for regular groups)
- AWD: Analog watchdog flag (set when an analog watchdog detection occurs)
- EOSMP: End of sampling flag (set when the end of sampling phase occurs)
- OVR: Overrun flag (set when the data overrun occurs)

Separate interrupt enable bits are available for flexibility.

Interrupt Event	Event Flag	Enable Control Bit
ADC ready	ADRDY	ADRDYIE
End of conversion of a regular channel	EOC	EOCIE
End of sequence of conversions of a regular group	EOS	EOSIE
Analog watchdog status bit is set	AWD	AWDIE
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE

12.5 Registers

12.5.1 Register Address Map

Offset	Register Name	Register Description
0x0000	ADC_ISR	Interrupt and status register
0x0004	ADC_IER	Interrupt enable register
0x0008	ADC_CR	ADC control register
0x000c	ADC_CFGR	ADC configuration register
0x0010	ADC_CFGR2	ADC configuration register 2
0x0014	ADC_SMPR1	ADC sample time register
0x0018	ADC_SMPR2	ADC sample time register
0x001c	ADC_AWDCR	Analog watchdog control register
0x0020	ADC_AWDTR	Analog watchdog threshold register
0x0024	ADC_SQR1	ADC sequence register 1
0x0028	ADC_SQR2	ADC sequence register 2
0x002c	ADC_SQR3	ADC sequence register 3
0x0030	ADC_SQR4	ADC sequence register 4
0x0034	ADC_DR	ADC data register
0x0038	ADC_DR0	ADC data register 0
0x003c	ADC_DR1	ADC data register 1
0x0040	ADC_DR2	ADC data register 2
0x0044	ADC_DR3	ADC data register 3

Offset	Register Name	Register Description
0x0048	ADC_DR4	ADC data register 4
0x004c	ADC_DR5	ADC data register 5
0x0050	ADC_DR6	ADC data register 6
0x0054	ADC_DR7	ADC data register 7
0x0058	ADC_DR8	ADC data register 8
0x005c	ADC_DR9	ADC data register 9
0x0060	ADC_DR10	ADC data register 10
0x0064	ADC_DR11	ADC data register 11
0x0068	ADC_DR12	ADC data register 12
0x006c	ADC_DR13	ADC data register 13
0x0070	ADC_DR14	ADC data register 14
0x0074	ADC_DR15	ADC data register 15
0x0078	ADC_DR16	ADC data register 16
0x007c	ADC_DR17	ADC data register 17
0x0080	ADC_DR18	ADC data register 18
0x0084	ADC_DR19	ADC data register 19
0x0088	ADC_OFR1	Offset register
0x008c	ADC_OFR2	Offset register
0x0090	ADC_OFR3	Offset register
0x0094	ADC_OFR4	Offset register
0x0098	ADC_DIFSEL	Differential channel select register

Offset	Register Name	Register Description
0x009c	ADC_CALFACT	Self calibration result register
0x00a0	ADC_GCOMP	Gain compensation register
0x00a4	ADC_CCR	ADC common control register
0x00a8	ADC_TSR	Temperature sensor calibration register

12.5.2 Register Field Details

12.5.2.1 ADC_ISR

0x0000			Interrupt and status register											ADC_ISR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	AWDHFG	AWDLFG	Reserved													
Type	RO	RO	RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										AWD	OVR	EOS	EOC	EOSMP	ADRDY
Type	RO										RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-3 Interrupt and Status Register Description

Field	Name	Description
31	AWDHFG	AWD high flag
30	AWDLFG	AWD low flag
29:6	Reserved	Reserved
5	AWD	Analog watchdog interrupt
4	OVR	ADC overrun interrupt
3	EOS	ADC end of sequence convert interrupt
2	EOC	ADC end of channel convert interrupt
1	EOSMP	ADC end of sample interrupt

Field	Name	Description
0	ADRDY	ADC ready interrupt

12.5.2.2 ADC_IER

0x0004			Interrupt enable register											ADC_IER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										AWDIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
Type	RO										RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-4 Interrupt Enable Register Description

Field	Name	Description
31:6	Reserved	Reserved
5	AWDIE	Analog watchdog interrupt enable, write this bit when ADSTART disable 0: Disable 1: Enable
4	OVRIE	ADC overrun interrupt enable, write this bit when ADSTART disable 0: Disable 1: Enable
3	EOSIE	ADC sequence convert done interrupt enable, write this bit when ADSTART disable 0: Disable 1: Enable
2	EOCIE	ADC channel convert done interrupt enable, write this bit when ADSTART disable 0: Disable

Field	Name	Description
		1: Enable
1	EOSMPIE	ADC end of sample interrupt enable, write this bit when ADSTART disable 0: Disable 1: Enable
0	ADRDYIE	ADC ready interrupt enable, write this bit when ADSTART disable 0: Disable 1: Enable

12.5.2.3 ADC_CR

0x0008			ADC control register											ADC_CR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	OSADCALI	GAINADCALI	ADCALIDIF	Reserved												
Type	RW	RW	RW	RO												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							BOOSTEN	Reserved				ADSTOP	ADSTART	ADDIS	ADCEN
Type	RO							RW	RO				RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-5 ADC Control Register Description

Field	Name	Description
31	OSADCALI	ADC offset calibration enable, write this bit when ADC disable 0: Disable 1: Enable
30	GAINADCALI	ADC gain calibration enable, write this bit when ADC disable 0: Disable 1: Enable
29	ADCALIDIF	ADC difference calibration enable, write this bit when ADC disable and calibration disable 0: Single 1: Difference
28:9	Reserved	Reserved
8	BOOSTEN	Boost enable

Field	Name	Description
		0: Disable 1: Enable
7:4	Reserved	Reserved
3	ADSTOP	ADC stop control bit
2	ADSTART	ADC start control bit, it is a software trigger or hardware trigger enable
1	ADDIS	ADC disable bit, clear dac enable
0	ADCEN	ADC enable

12.5.2.4 ADC_CFGR

0x000c			ADC configuration register											ADC_CFGR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved											DISCNUM			DISCEN	ALIGN
Type	RO											RW			RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	AUTOOFF	AUTODLY	CONT	OVRMOD	EXTEN		EXTSEL					RES		Reserved		DMAEN
Type	RW	RW	RW	RW	RW		RW					RW		RO		RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-6 ADC Configuration Register Description

Field	Name	Description
31:21	Reserved	Reserved
20:18	DISCNUM	N channel converted after once triggered in discontinuous mode, write these bits when ADSTART = 0 3'h0: 1 channel
17	DISCEN	Discontinuous mode enable, cannot configure discontinuous mode and continuous mode at the same time; write these bits when ADSTART = 0
16	ALIGN	Data align, write these bits when ADSTART = 0 0: Right 1: Left
15	AUTOOFF	Auto off mode enable, write these bits when ADSTART = 0

Field	Name	Description
14	AUTODLY	Auto delay mode enable, write these bits when ADSTART = 0
13	CONT	Continuous mode enable, write these bits when ADSTART = 0
12	OVRMOD	Overrun over write enable, write these bits when ADSTART = 0 0: Disable write data when overrun 1: Overwrite data when overrun
11:10	EXTEN	Trigger polar select, write these bits when ADSTART = 0 00: Software trigger 01: Hardware posedge trigger 10: Hardware negedge trigger 11: Hardware posedge trigger or negedge trigger
9:5	EXTSEL	External trigger channel select, write these bits when ADSTART = 0 0x00: RTC wakeup trigger 0x01: cmp0 trigger 0x02: cmp1 trigger 0x03: gen_timer0_otrg trigger 0x04: gen_timer0_cc_int trigger 0x05: gen_timer1_otrg trigger 0x06: gen_timer1_cc_int trigger 0x07: gen_timer2_otrg trigger 0x08: gen_timer2_cc_int trigger 0x09: gen_timer3_otrg trigger 0x0A: gen_timer3_cc_int trigger 0x0B: gen_timer4_otrg trigger 0x0C: gen_timer4_cc_int trigger 0x0D: gen_timer5_otrg trigger 0x0E: gen_timer5_cc_int trigger 0x0F: exti15 trigger

Field	Name	Description
		0x10: exti2 trigger 0x11: adv_timer0_otrg trigger 0x12: adv_timer0_otrg2 trigger 0x13: adv_timer0_cc_int trigger 0x14: adv_timer1_otrg trigger 0x15: adv_timer1_otrg2 trigger 0x16: adv_timer1_cc_int trigger 0x17: adv_timer2_otrg trigger 0x18: adv_timer2_otrg2 trigger 0x19: adv_timer2_cc_int trigger
4:3	RES	Resolution, write these bits when ADSTART = 0 00: 12-bit 01: 10-bit 10: 8-bit
2:1	Reserved	Reserved
0	DMAEN	DMA enable, write this bit when ADSTART = 0 0: Disable 1: Enable

12.5.2.5 ADC_CFGR2

0x0010			ADC configuration register 2											ADC_CFGR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved													SMPPLUS	Reserved	
Type	RO													RW	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						OVSMPTEN	OVSS				Reserved	OVSR			OVSMPEN
Type	RO						RW	RW				RO	RW			RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-7 ADC Configuration Register 2 Description

Field	Name	Description
31:19	Reserved	Reserved
18	SMPPLUS	Sample time increase one cycle enable, write this bit when ADSTART = 0
17:10	Reserved	Reserved
9	OVSMPTEN	Over sample trigger mode
8:5	OVSS	Over sample right shift bits, write these bits when ADSTART = 0 0000: No shift 0001: Shift 1-bit 0010: Shift 2-bit 0011: Shift 3-bit 0100: Shift 4-bit 0101: Shift 5-bit

Field	Name	Description
		0110: Shift 6-bit 0111: Shift 7-bit 1000: Shift 8-bit
4	Reserved	Reserved
3:1	OVSR	Over sample rate, write these bits when ADSTART = 0 000: 2x 001: 4x 010: 8x 011: 16x 100: 32x 101: 64x 110: 128x 111: 256x
0	OVSMPEN	Over sample enable, write this bit when ADSTART = 0

12.5.2.6 ADC_SMPR1

0x0014			ADC sample time register											ADC_SMPR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		SMP9			SMP8			SMP7			SMP6			SMP5	
Type	RO		RW			RW			RW			RW			RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SMP5	SMP4			SMP3			SMP2			SMP1			SMP0		
Type	RW	RW			RW			RW			RW			RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-8 ADC Sample Time Register Description

Field	Name	Description
31:30	Reserved	Reserved
29:27	SMP9	Refer to SMP0 description
26:24	SMP8	Refer to SMP0 description
23:21	SMP7	Refer to SMP0 description
20:18	SMP6	Refer to SMP0 description
17:15	SMP5	Refer to SMP0 description
14:12	SMP4	Refer to SMP0 description
11:9	SMP3	Refer to SMP0 description
8:6	SMP2	Refer to SMP0 description
5:3	SMP1	Refer to SMP0 description

Field	Name	Description
2:0	SMP0	Channel 0 sample time control bits, write these bits when ADSTART = 0 000: 3 cycles 001: 5 cycles 010: 11 cycles 011: 25 cycles 100: 45 cycles 101: 165 cycles 110: 345 cycles 111: 705 cycles

12.5.2.7 ADC_SMPR2

0x0018			ADC sample time register											ADC_SMPR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		SMP19			SMP18			SMP17			SMP16			SMP15	
Type	RO		RW			RW			RW			RW			RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SMP15	SMP14			SMP13			SMP12			SMP11			SMP10		
Type	RW	RW			RW			RW			RW			RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-9 ADC Sample Time Register Description

Field	Name	Description
31:30	Reserved	Reserved
29:27	SMP19	Refer to SMP0 description
26:24	SMP18	Refer to SMP0 description
23:21	SMP17	Refer to SMP0 description
20:18	SMP16	Refer to SMP0 description
17:15	SMP15	Refer to SMP0 description
14:12	SMP14	Refer to SMP0 description
11:9	SMP13	Refer to SMP0 description
8:6	SMP12	Refer to SMP0 description
5:3	SMP11	Refer to SMP0 description

Field	Name	Description
2:0	SMP10	Refer to SMP0 description

12.5.2.8 ADC_AWDCR

0x001c			Analog watchdog control register											ADC_AWDCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved												AWDCH			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	AWDCH															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-10 Analog Watchdog Control Register Description

Field	Name	Description
31:20	Reserved	Reserved
19:0	AWDCH	Analog watchdog channel enable, write these bits when ADSTART = 0

12.5.2.9 ADC_AWDTR

0x0020			Analog watchdog threshold register											ADC_AWDTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				AWDD MAEN	AWDFILT			HTH							
Type	RO				RW	RW			RW							
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	HTH				LTH											
Type	RW				RW											
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-11 Analog Watchdog Threshold Register Description

Field	Name	Description
31:28	Reserved	Reserved
27	AWDDMAEN	Analog watchdog support DMA enable, write this bit when ADSTART = 0
26:24	AWDFILT	Analog watch filter number, valid monitor single channel, write these bits when ADSTART = 0 000: 1 001: 2 010: 3 011: 4 100: 5 101: 6 110: 7 111: 8

Field	Name	Description
23:12	HTH	Analog watchdog high threshold, write these bits when ADSTART = 0
11:0	LTH	Analog watchdog low threshold, write these bits when ADSTART = 0

12.5.2.10 ADC_SQR1

0x0024			ADC sequence register 1											ADC_SQR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		SQ3						Reserved	SQ2					Reserved	SQ1
Type	RO		RW						RO	RW					RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SQ1				Reserved	SQ0					Reserved		CHLENGTH			
Type	RW				RO	RW					RO		RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-12 ADC Sequence Register 1 Description

Field	Name	Description
31:29	Reserved	Reserved
28:24	SQ3	ADC channel index, write these bits when ADSTART = 0
23	Reserved	Reserved
22:18	SQ2	ADC channel index, write these bits when ADSTART = 0
17	Reserved	Reserved
16:12	SQ1	ADC channel index, write these bits when ADSTART = 0
11	Reserved	Reserved
10:6	SQ0	ADC channel index, write these bits when ADSTART = 0
5:4	Reserved	Reserved

Field	Name	Description
3:0	CHLENGTH	ADC channel length, write these bits when ADSTART = 0

12.5.2.11 ADC_SQR2

0x0028			ADC sequence register 2											ADC_SQR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		SQ8						Reserved	SQ7					Reserved	SQ6
Type	RO		RW						RO	RW					RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SQ6				Reserved	SQ5					Reserved	SQ4				
Type	RW				RO	RW					RO	RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-13 ADC Sequence Register 2 Description

Field	Name	Description
31:29	Reserved	Reserved
28:24	SQ8	ADC channel index, write these bits when ADSTART = 0
23	Reserved	Reserved
22:18	SQ7	ADC channel index, write these bits when ADSTART = 0
17	Reserved	Reserved
16:12	SQ6	ADC channel index,write these bits when ADSTART = 0
11	Reserved	Reserved
10:6	SQ5	ADC channel index, write these bits when ADSTART = 0
5	Reserved	Reserved

Field	Name	Description
4:0	SQ4	ADC channel index, write these bits when ADSTART = 0

12.5.2.12 ADC_SQR3

0x002c			ADC sequence register 3											ADC_SQR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		SQ13						Reserved	SQ12					Reserved	SQ11
Type	RO		RW						RO	RW					RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SQ11				Reserved	SQ10					Reserved	SQ9				
Type	RW				RO	RW					RO	RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-14 ADC Sequence Register 3 Description

Field	Name	Description
31:29	Reserved	Reserved
28:24	SQ13	ADC channel index, write these bits when ADSTART = 0
23	Reserved	Reserved
22:18	SQ12	ADC channel index, write these bits when ADSTART = 0
17:17	Reserved	Reserved
16:12	SQ11	ADC channel index, write these bits when ADSTART = 0
11	Reserved	Reserved
10:6	SQ10	ADC channel index,write these bits when ADSTART = 0
5	Reserved	Reserved

Field	Name	Description
4:0	SQ9	ADC channel index,write these bits when ADSTART = 0

12.5.2.13 ADC_SQR4

0x0030			ADC sequence register 4											ADC_SQR4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					SQ15					Reserved	SQ14				
Type	RO					RW					RO	RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-15 ADC Sequence Register 4 Description

Field	Name	Description
31:11	Reserved	Reserved
10:6	SQ15	ADC channel index,write these bits when ADSTART = 0
5	Reserved	Reserved
4:0	SQ14	ADC channel index,write these bits when ADSTART = 0

12.5.2.14 ADC_DR

0x0034			ADC data register											ADC_DR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-16 ADC Data Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA	ADC convert result

12.5.2.15 ADC_DR0

0x0038			ADC data register 0											ADC_DR0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA0															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-17 ADC Data Register 0 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA0	ADC channel 0 convert result

12.5.2.16 ADC_DR1

0x003c			ADC data register 1											ADC_DR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA1															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-18 ADC Data Register 1 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA1	ADC channel 1 convert result

12.5.2.17 ADC_DR2

0x0040			ADC data register 2											ADC_DR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA2															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-19 ADC Data Register 2 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA2	ADC channel 2 convert result

12.5.2.18 ADC_DR3

0x0044			ADC data register 3											ADC_DR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA3															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-20 ADC Data Register 3 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA3	ADC channel 3 convert result

12.5.2.19 ADC_DR4

0x0048			ADC data register 4											ADC_DR4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA4															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-21 ADC Data Register 4 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA4	ADC channel 4 convert result

12.5.2.20 ADC_DR5

0x004c			ADC data register 5											ADC_DR5		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA5															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-22 ADC Data Register 5 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA5	ADC channel 5 convert result

12.5.2.21 ADC_DR6

0x0050			ADC data register 6											ADC_DR6		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA6															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-23 ADC Data Register 6 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA6	ADC channel 6 convert result

12.5.2.22 ADC_DR7

0x0054			ADC data register 7											ADC_DR7		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA7															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-24 ADC Data Register 7 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA7	ADC channel 7 convert result

12.5.2.23 ADC_DR8

0x0058			ADC data register 8											ADC_DR8		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA8															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-25 ADC Data Register 8 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA8	ADC channel 8 convert result

12.5.2.24 ADC_DR9

0x005c			ADC data register 9											ADC_DR9		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA9															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-26 ADC Data Register 9 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA9	ADC channel 9 convert result

12.5.2.25 ADC_DR10

0x0060			ADC data register 10											ADC_DR10		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA10															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-27 ADC Data Register 10 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA10	ADC channel 10 convert result

12.5.2.26 ADC_DR11

0x0064			ADC data register 11											ADC_DR11		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA11															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-28 ADC Data Register 11 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA11	ADC channel 11 convert result

12.5.2.27 ADC_DR12

0x0068			ADC data register 12											ADC_DR12		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA12															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-29 ADC Data Register 12 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA12	ADC channel 12 convert result

12.5.2.28 ADC_DR13

0x006c			ADC data register 13											ADC_DR13		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA13															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-30 ADC Data Register 13 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA13	ADC channel 13 convert result

12.5.2.29 ADC_DR14

0x0070			ADC data register 14											ADC_DR14		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA14															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-31 ADC Data Register 14 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA14	ADC channel 14 convert result

12.5.2.30 ADC_DR15

0x0074			ADC data register 15											ADC_DR15		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA15															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-32 ADC Data Register 15 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA15	ADC channel 15 convert result

12.5.2.31 ADC_DR16

0x0078			ADC data register 16											ADC_DR16		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA16															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-33 ADC Data Register 16 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA16	ADC channel 16 convert result

12.5.2.32 ADC_DR17

0x007c			ADC data register 17											ADC_DR17		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA17															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-34 ADC Data Register 17 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA17	ADC channel 17 convert result

12.5.2.33 ADC_DR18

0x0080			ADC data register 18											ADC_DR18		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA18															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-35 ADC Data Register 18 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA18	ADC channel 18 convert result

12.5.2.34 ADC_DR19

0x0084			ADC data register 19											ADC_DR19		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CONVDATA19															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-36 ADC Data Register 19 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CONVDATA19	ADC channel 19 convert result

12.5.2.35 ADC_OFR1

0x0088			Offset register											ADC_OFR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	OFFSET1EN	OFFSET1CH					SATEN	OFFSETPOS	Reserved							
Type	RW	RW					RW	RW	RO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				OFFSET1											
Type	RO				RW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-37 Offset Register Description

Field	Name	Description
31	OFFSET1EN	Offset calculate enable, write this bit when ADSTART = 0
30:26	OFFSET1CH	Offset channel index, write these bits when ADSTART = 0
25	SATEN	Data saturation enable, write this bit when ADSTART = 0
24	OFFSETPOS	Offset positive enable, write this bit when ADSTART = 0 0: Negative 1: Positive
23:12	Reserved	Reserved
11:0	OFFSET1	Offset value, write these bits when ADSTART = 0

12.5.2.36 ADC_OFR2

0x008c			Offset register											ADC_OFR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	OFFSET2EN	OFFSET2CH					Reserved									
Type	RW	RW					RO									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				OFFSET2											
Type	RO				RW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-38 Offset Register Description

Field	Name	Description
31	OFFSET2EN	Offset calculate enable, write this bit when ADSTART = 0
30:26	OFFSET2CH	Offset channel index, write these bits when ADSTART = 0
25:12	Reserved	Reserved
11:0	OFFSET2	Offset value, write these bits when ADSTART = 0

12.5.2.37 ADC_OFR3

0x0090			Offset register											ADC_OFR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	OFFSET3EN	OFFSET3CH					Reserved									
Type	RW	RW					RO									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				OFFSET3											
Type	RO				RW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-39 Offset Register Description

Field	Name	Description
31	OFFSET3EN	Offset calculate enable, write this bit when ADSTART = 0
30:26	OFFSET3CH	Offset channel index, write these bits when ADSTART = 0
25:12	Reserved	Reserved
11:0	OFFSET3	Offset value, write these bits when ADSTART = 0

12.5.2.38 ADC_OFR4

0x0094			Offset register											ADC_OFR4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	OFFSET4EN	OFFSET4CH					Reserved									
Type	RW	RW					RO									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				OFFSET4											
Type	RO				RW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-40 Offset Register Description

Field	Name	Description
31	OFFSET4EN	Offset calculate enable, write this bit when ADSTART = 0
30:26	OFFSET4CH	Offset channel index, write these bits when ADSTART = 0
25:12	Reserved	Reserved
11:0	OFFSET4	Offset value, write these bits when ADSTART = 0

12.5.2.39 ADC_DIFSEL

0x0098			Differential channel select register											ADC_DIFSEL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved												DIFSEL			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DIFSEL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-41 Differential Channel Select Register Description

Field	Name	Description
31:20	Reserved	Reserved
19:0	DIFSEL	Differential channel enable,write these bits when ADC disable

12.5.2.40 ADC_CALFACT

0x009c			Self calibration result register											ADC_CALFACT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved						GAINTRIMS					GAINTRIMD				
Type	RO						RW					RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						OSTRIMS					OSTRIMD				
Type	RO						RW					RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-42 Self Calibration Result Register Description

Field	Name	Description
31:26	Reserved	Reserved
25:21	GAINTRIMS	Single channel gain calibration value, write these bits when ADSTART = 0 These bits are trimming data, default data is 5'hX
20:16	GAINTRIMD	Differential channel gain calibration value, write these bits when ADSTART = 0 These bits are trimming data, default data is 5'hX
15:10	Reserved	Reserved
9:5	OSTRIMS	Single channel offset calibration value, write these bits when ADSTART = 0 These bits are trimming data, default data is 5'hX
4:0	OSTRIMD	Differential channel offset calibration value, write these bits when ADSTART = 0 These bits are trimming data, default data is 5'hX

12.5.2.41 ADC_GCOMP

0x00a0			Gain compensation register											ADC_GCOMP		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															GAINEN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		GAINCOEF													
Type	RO		RW													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-43 Gain Compensation Register Description

Field	Name	Description
31:17	Reserved	Reserved
16	GAINEN	Convert result gain compensation enable,write these bits when ADSTART = 0
15:14	Reserved	Reserved
13:0	GAINCOEF	Gain compensation value, write these bits when ADSTART = 0

12.5.2.42 ADC_CCR

0x00a4			ADC common control register											ADC_CCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				HSMODE	VREFSEL	HALFVDDEN	VSENSEEN	VBATEN	Reserved	PRESC				CKMODE	
Type	RO				RW	RW	RW	RW	RW	RO	RW				RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-44 ADC Common Control Register Description

Field	Name	Description
31:12	Reserved	Reserved
11	HSMODE	ADC high speed mode enable, configure this bit when fclk > 24M; write this bit when ADC disable
10	VREFSEL	Reference voltage select, write this bit when ADC disable 0: Vref+/Vref- 1: VDD/VSS
9	HALFVDDEN	Half vdd channel enable, write this bit when ADC disable
8	VSENSEEN	Temperature enable, write this bit when ADC disable
7	VBATEN	VBAT/2 channel enable, write this bit when ADC disable
6	Reserved	Reserved

Field	Name	Description
5:2	PRESC	adc_pclk divider coefficient, write this bit when ADC disable 0000: 1 0001: 2 0010: 4 0011: 6 0100 8 0101: 10 0110: 12 0111: 16 1000: 32 1001: 64 1010: 128 1011: 256
1:0	CKMODE	CLK select, write this bit when ADC disable 00: adc_pclk 01: adc_hclk/1 10: adc_hclk/2 11: adc_hclk/4

12.5.2.43 ADC_TSR

0x00a8			Temperature sensor calibration register											ADC_TSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				TS125C											
Type	RO				RO											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				TS25C											
Type	RO				RO											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-45 Temperature Sensor Calibration Register Description

Field	Name	Description
31:28	Reserved	Reserved
27:16	TS125C	Temperature sensor calibration value in 125℃
15:12	Reserved	Reserved
11:0	TS25C	Temperature sensor calibration value in 25℃

Digital-to-Analog Converter (DAC12)

This chapter provides a detailed description of the Digital-to-Analog Converter (DAC12), covering its components, functionality, and operation in depth.

Topics: **Page**

13.1 Introduction

The Digital-to-Analog Converter (DAC) module is a 12-bit voltage-output digital-to-analog converter. It can be configured in either 8-bit or 12-bit mode. The DAC module supports up to two output channels. More details about the DAC module are described in the following sections.

13.2 Features

- Left or right data alignment in 12-bit/8-bit mode
- Data preload function via FIFO with a depth of 4
- Generation of noise wave and triangular wave
- One DAC core for independent dual output channels
- DMA capability for efficient data transfer
- Support for external triggers or software trigger for conversion
- DAC output channel buffered or unbuffered modes
- Buffer offset calibration for precise output
- Option to disconnect DAC output from the DAC_OUTy output pin
- DAC output connection to on-chip peripherals
- Input voltage reference V_{REF+}

13.3 Functional Description

13.3.1 Block Diagram

Figure 13-1 shows the DAC block diagram.

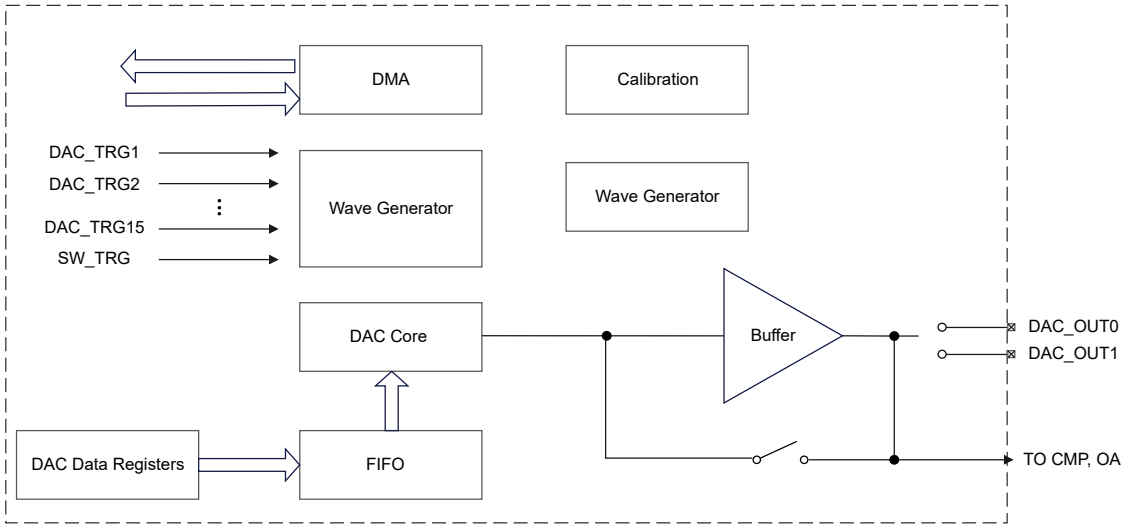


Figure 13-1 DAC Block Diagram

13.3.2 DAC Pins

13.3.2.1 DAC Reference

The reference for the DAC is sourced from V_{REF}, and can be connected to the signal as below:

- VREFBUF when VREFBUF is enabled
- V_{DDA} via V_{REF+} pin when VREFBUF is disabled
- V_{REF} via V_{REF+} pin when VREFBUF is disabled

13.3.2.2 DAC Output

The DAC has a single core with dual output channels, which can be selected using the CHSEL bit.

- When CHSEL = 0, DAC_OUT0 is selected, DAC_OUT1 is high-z.
- When CHSEL = 1, DAC_OUT1 is selected, DAC_OUT0 is high-z.

Figure 13-2 shows the details of DAC output.

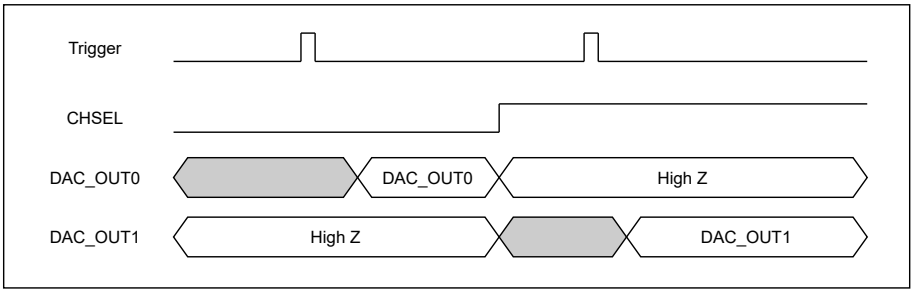


Figure 13-2 DAC Output Example

NOTE: When the DAC outputs to an external pad, the buffer should be enabled.

DAC can also be output to internal modules with a buffer enable or disable option. For more details, refer to Table 13-1.

Table 13-1 DAC Output to Internal Modules with Buffer

Mode	Buffer	Internal	External
0	Enabled	No	Yes
1	Enabled	Yes	Yes
2	Reserved	Reserved	Reserved
3	Disabled	Yes	No

13.3.2.3 DAC Triggers

Refer to device specific for DAC trigger signal details.

13.3.2.4 DAC DMA

The DAC module has a DMA capability and supports single and double data modes.

- In single data mode, a DMA request transfers only 12-bit (or 8-bit) data.
- In the double data mode, two 12-bit data are transferred simultaneously. The DMADOUBLE bit of the DAC_MCR register is used to control this mode.

Figure 13-3 shows the block diagram of the single data mode.

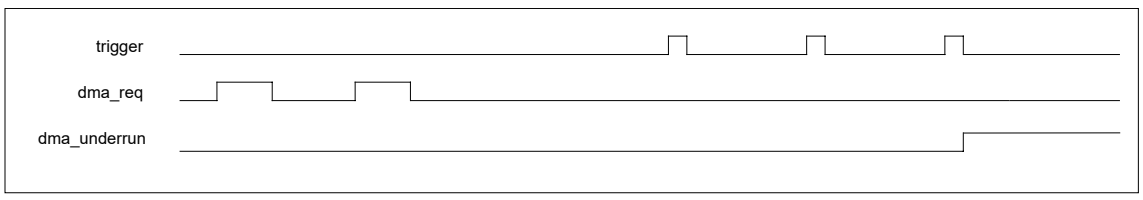


Figure 13-3 Single Data Mode Block Diagram

Figure 13-4 shows the block diagram of the double data mode.

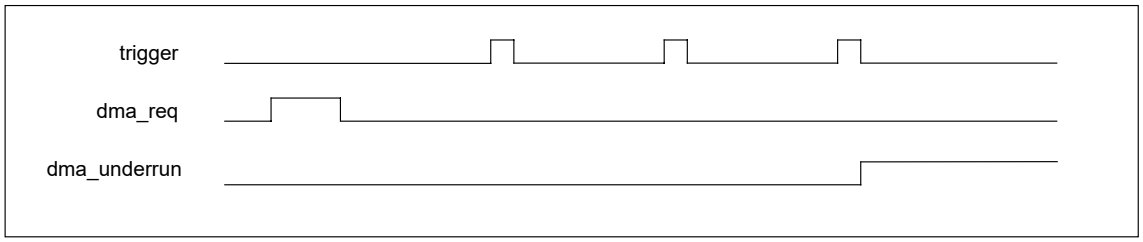


Figure 13-4 Double Data Mode Block Diagram

DMA Underrun

The DAC DMA request is not queued to prevent the issuance of a new request if a second external trigger arrives before the acknowledgement for the first external trigger is received (first request). In this case, the DMA underrun flag (DMAUDR) in the DAC_SR register is set, reporting the error condition. The DMA continues to convert the old data.

To address this error condition and restart the transfer correctly, the software needs to perform the following steps:

1. Clear the DMAUDR flag by writing 1 to it.
2. Clear the DMAEN bit of the used DMA stream.
3. Reinitialize both the DMA and DAC.

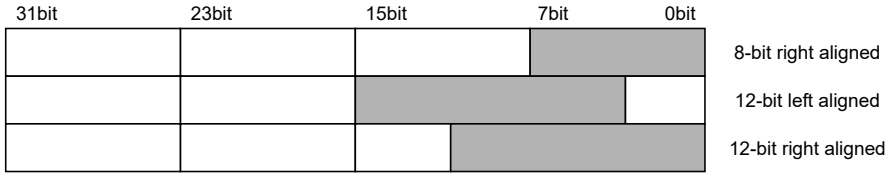
To avoid new DMA underruns, the software can modify the DAC trigger conversion frequency or reduce the DMA workload.

An interrupt is also generated if its corresponding DMAUDRIE bit in the DAC_CR register is enabled. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

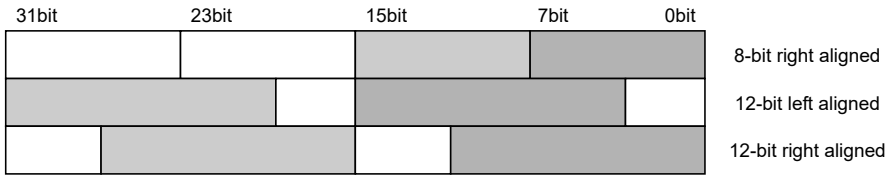
13.3.3 DAC Data Format

The DAC data format can be configured as either right-aligned or left-aligned.

Single data format:



Double data format:



Signed/Unsigned Data

The input data for DAC is typically unsigned, such as in the case of a 12-bit DAC: 0x000 represents the minimum value, while 0xFFF represents the maximum value.

Additionally, the DAC can handle signed data, which is controlled by the SINFORMAT bit in the DAC_MCR register (refer to Table 13-2).

Table 13-2 12-Bit Signed Data Format

SINFORMAT	DATA Written to DHR Register	DATA Transferred to DOR Register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

13.3.4 DAC FIFO

The DAC input data supports FIFO mode, enabling continuous data writing. The FIFO has a bit width of 12 and a depth of 4, and can handle 4 single mode or 2 double mode transmissions.

Data is written to the FIFO in single data mode, with the least significant 12 bits of the DHR data being written into the FIFO. In double data mode, the FIFO is written following the steps below:

1. The data from the least significant side of the DHR ([11:0], [15:4], or [7:0]) is written into the FIFO.
2. The data from the most significant side of the DHR ([27:16], [31:20], or [15:8]) is written into the FIFO.

When the trigger event occurs, the preload data will be transferred to the DAC core and conversion on TRG_EN = 1.

Figure 13-5 shows the block diagram of FIFO control.

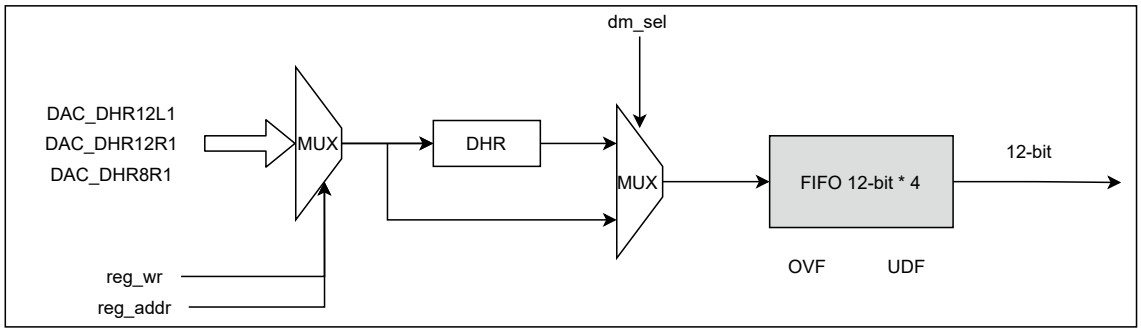


Figure 13-5 Block Diagram of FIFO Control

13.3.5 DAC Signal Generator

Generating Noise

A linear feedback shift register (LFSR) generates a pseudonoise with variable amplitude. To enable DAC noise generation, set the WAVE to 2'b01 and reset it by resetting this bit. The preloaded value in the LFSR is 0xAAA.

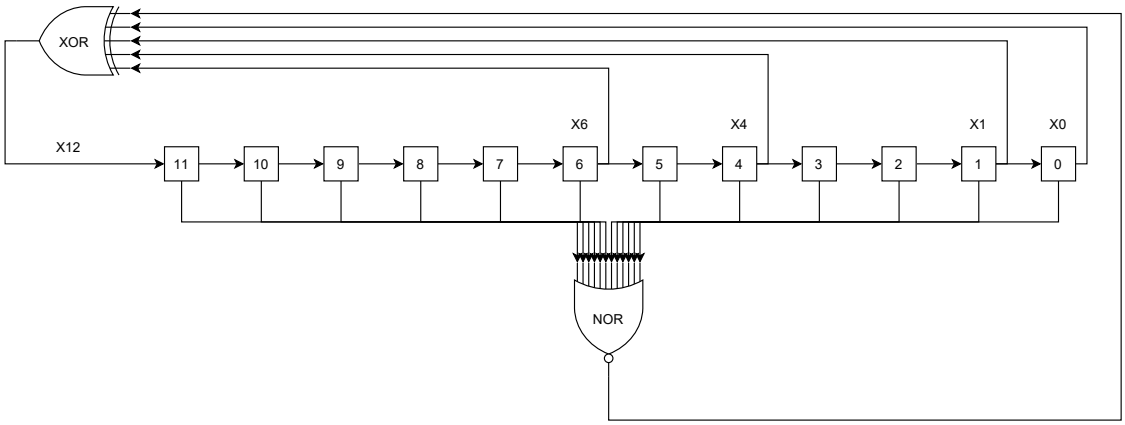


Figure 13-6 Calculation Algorithm for DAC LFSR Register

Generating Triangle Waves

To generate a triangle wave using the DAC, follow these steps:

1. Enable triangle-wave generation by setting the WAVE to 2'b10.
2. Use the MAMP bits to configure the desired amplitude.
3. After each trigger event, increment an internal triangle counter.
4. Add the value of this counter to the DAC_DHRx register without causing overflow.
5. Transfer the sum into the DAC_DORx register.
6. Continue incrementing the triangle counter as long as it remains less than the maximum amplitude defined by the MAMPx[3:0] bits.
7. Once the configured amplitude is reached, decrement the counter down to 0 and then increment it again.

   Repeat this process to create a continuous triangle wave pattern.

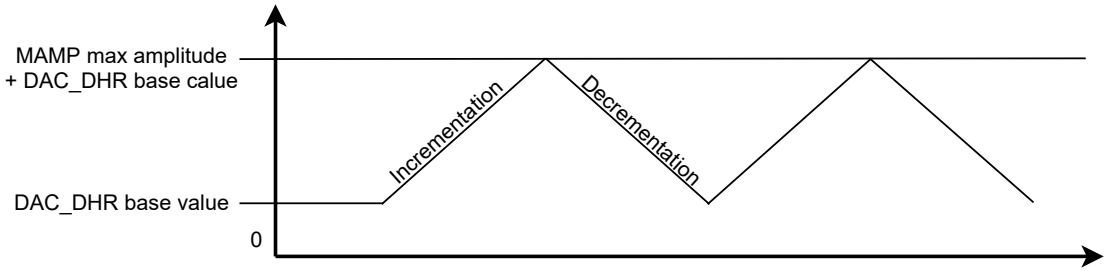


Figure 13-7 DAC Triangle Wave Generation

13.3.6 DAC Core

When the DAC output is enabled (DACEN = 1), the maximum value for DAC_DHRx is 0FFFh, which leads to a rail output of (4095 ÷ 4096) × $V_{REF.}$ The minimum value for DAC_DHRx is 0000h, which results in a ground output. A value greater than 4095 can be written to the register, but all leading bits are ignored.

Resolution	Output Voltage
12-bit	$V_{out} = V_{ref}$ x DAC_DHRx / 4096

13.3.7 DAC Buffer Calibration

Due to variations in the output buffer characteristics, the voltage offset may differ between different parts, leading to an absolute offset error in the analog output. To compensate for Vos, a calibration is required using a trimming technique.

The following are the steps for calibration:

1. Enable the CEN and DACEN bits.
2. Write the value 0x0000 to the register OFFSET[4:0] and start the calibration.
3. If CALFLAG = 0 (or 1), write the value 0x00001 to the register OFFSET[4:0].
4. Repeat the flow untill CALFLAG = 1 (or 0), and calibration is done.

13.4 Interrupts

The DAC module can generate an interrupt when the interrupt enable (IE) bits are set. The FIFOOVF or FIFOUDF will be set when the DAC data FIFO overflows or underflows. In DMA mode, if an underrun occurs, the DMAUDR will be set.

Interrupt Enable (IE)	Interrupt Enable (IE) Flag	Description
FIFOOVFIE	FIFOOVF	DAC data FIFO overflow interrupt
FIFOUDFIE	FIFOUDF	DAC data FIFO underflow interrupt
DMAUDRIE	DMAUDR	DAC DMA underrun interrupt

13.5 Registers

13.5.1 Register Address Map

Offset	Register Name	Register Description
0x0000	DAC_CR	Control register
0x0004	DAC_SWTRGR	Software trigger register
0x0008	DAC_DHR12R	Double data mode 12-bit right align register
0x000c	DAC_DHR12L	12-bit left align register
0x0010	DAC_DHR8R	8-bit right align register
0x0014	DAC_DOR	Data output register
0x0018	DAC_SR	Status register
0x001c	DAC_MCR	Mode control register
0x0020	DAC_CCR	Calibration register

13.5.2 Register Field Details

13.5.2.1 DAC_CR

0x0000			Control register											DAC_CR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	CEN	DMAUDRIE	DMAEN	MAMP				WAVE		TSEL				TEN	EN
Type	RO	RW	RW	RW	RW				RW		RW				RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-3 Control Register Description

Field	Name	Description
31:15	Reserved	Reserved
14	CEN	DAC calibration enable 0: Disable 1: Enable
13	DMAUDRIE	DAC DMA underrun interrupt enable 0: Disable 1: Enable
12	DMAEN	DAC DMA enable 0: Disable 1: Enable

Field	Name	Description
11:8	MAMP	DAC LFSR mask/triangle amplitude; configure these bits before DAC enable 0000: Unmask bit[0]/triangle amplitude = 1 0001: Unmask bit[1:0]/triangle amplitude = 3 0010: Unmask bit[2:0]/triangle amplitude = 7 0011: Unmask bit[3:0]/triangle amplitude = 15 0100: Unmask bit[4:0]/triangle amplitude = 31 0101: Unmask bit[5:0]/triangle amplitude = 63 0110: Unmask bit[6:0]/triangle amplitude = 127 0111: Unmask bit[7:0]/triangle amplitude = 255 1000: Unmask bit[8:0]/triangle amplitude = 511 1001: Unmask bit[9:0]/triangle amplitude = 1023 1010: Unmask bit[10:0]/triangle amplitude = 2047 ≥1011: Unmask bit[11:0]/triangle amplitude = 4095
7:6	WAVE	Noise waveform control bits 00: Disable noise waveform 01: Enable LSFR noise waveform 10: Enable triangle waveform (only valid when TEN = 0)
5:2	TSEL	Trigger select bits, configure these bits before DAC enable 0000: Software trigger select 0001: gen_timer0_otrg trigger select 0010: gen_timer1_otrg trigger select 0011: gen_timer2_otrg trigger select 0100: gen_timer3_otrg trigger select 0101: gen_timer4_otrg trigger select 0110: gen_timer5_otrg trigger select 0111: adv_timer0_otrg trigger select 1000: adv_timer1_otrg trigger select

Field	Name	Description
		1001: adv_timer2_otrg trigger select 1011: exit[4] trigger select
1	TEN	DAC trigger enable; set this bit after trigger select bits are configured. 0: Disable 1: Enable
0	EN	DAC enable

13.5.2.2 DAC_SWTRGR

0x0004			Software trigger register											DAC_SWTRGR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															SWTRIG
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-4 Software Trigger Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	SWTRIG	DAC software trigger, cleared by hardware

13.5.2.3 DAC_DHR12R

0x0008			Double data mode 12-bit right align register											DAC_DHR12R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				DACDHRB											
Type	RO				RW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				DACDHR											
Type	RO				RW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-5 Double Data Mode 12-Bit Right Align Register Description

Field	Name	Description
31:28	Reserved	Reserved
27:16	DACDHRB	DAC double data mode 12-bit right align register, double mode valid
15:12	Reserved	Reserved
11:0	DACDHR	DAC double data mode 12-bit right align register

13.5.2.4 DAC_DHR12L

0x000c			12-bit left align register										DAC_DHR12L			
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DACDHRB												Reserved			
Type	RW												RO			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DACDHR												Reserved			
Type	RW												RO			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-6 12-Bit Left Align Register Description

Field	Name	Description
31:20	DACDHRB	DAC double data mode 12-bit left align register, double mode valid
19:16	Reserved	Reserved
15:4	DACDHR	DAC double data mode 12-bit left align register

13.5.2.5 DAC_DHR8R

0x0010			8-bit right align register											DAC_DHR8R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DACDHRB								DACDHR							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-7 8-Bit Right Align Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	DACDHRB	DAC double data mode 8-bit right align register, double mode valid
7:0	DACDHR	DAC double data mode 8-bit right align register

13.5.2.6 DAC_DOR

0x0014			Data output register											DAC_DOR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				DACDORB											
Type	RO				RO											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				DACDOR											
Type	RO				RO											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-8 Data Output Register Description

Field	Name	Description
31:28	Reserved	Reserved
27:16	DACDORB	DAC data output register, double mode valid
15:12	Reserved	Reserved
11:0	DACDOR	DAC data output register

13.5.2.7 DAC_SR

0x0018			Status register											DAC_SR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	CALFLAG	DMAUDR	DORSTATUS	FIFODEPTH			FIFOOVF	FIFOUDF	FIFOEMPTY	FIFOFULL	Reserved				
Type	RO	RO	RC_W1	RO	RO			RC_W1	RC_W1	RO	RO	RO				
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 13-9 Status Register Description

Field	Name	Description
31:15	Reserved	Reserved
14	CALFLAG	Calibration flag, calibartion done when this bit reversed.
13	DMAUDR	DAC DMA underrun flag
12	DORSTATUS	DAC double data mode flag, valid in double mode 0: Low range bits flag 1: High range bits flag
11:9	FIFODEPTH	Number of data to be convert in the DAC FIFO
8	FIFOOVF	DAC data FIFO overflow flag
7	FIFOUDF	DAC data FIFO underflow flag
6	FIFOEMPTY	DAC data FIFO empty flag

Field	Name	Description
5	FIFOFULL	DAC data FIFO full flag
4:0	Reserved	Reserved

13.5.2.8 DAC_MCR

0x001c			Mode control register											DAC_MCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						SINFORMAT	DMADOUBLE	FIFOOVFIE	FIFOUDFIE	Reserved		CHSEL	Reserved	MODE	
Type	RO						RW	RW	RW	RW	RO		RW	RO	RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Table 13-10 Mode Control Register Description

Field	Name	Description
31:10	Reserved	Reserved
9	SINFORMAT	Enable signed format for DAC This bit is set and cleared by software. 0: Input data is in unsigned format 1: Input data is in signed format (2's complement). The MSB bit represents the sign.
8	DMADOUBLE	DAC DMA double data mode enable; configure this bit before DAC and DMA enable 0: Disable 1: Enable
7	FIFOOVFIE	DAC data FIFO overflow interrupt
6	FIFOUDFIE	DAC data FIFO underflow interrupt
5:4	Reserved	Reserved

Field	Name	Description
3	CHSEL	DAC output channel selection 0: DAC output channel 0 selected 1: DAC output channel 1 selected
2:2	Reserved	Reserved
1:0	MODE	DAC work mode, can not write these bits when DAC enable or calibration enable 00: External pin + Buffer enable 01: External pin and chip + Buffer enable 10: Reserved 11: On-chip + Buffer disable

13.5.2.9 DAC_CCR

0x0020			Calibration register											DAC_CCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved											OFFSET				
Type	RO											RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-11 Calibration Register Description

Field	Name	Description
31:5	Reserved	Reserved
4:0	OFFSET	DAC buffer calibration register These bits are trimming data, the default value is 5'hX.

True Random Number Generator (TRNG)

This chapter describes the details of the True Random Number Generator (TRNG).

Topics: **Page**

14.1 Introduction

The True Random Number Generator (TRNG) generates a random bit stream. It is commonly used for key generation or to seed approved deterministic random number generators.

The TRNG produces unpredictable and unbiased random numbers, making it highly secure and reliable for cryptographic applications. With its ability to generate truly random numbers, it ensures the integrity and confidentiality of sensitive data.

Additionally, the TRNG's random bit stream serves as a valuable source of entropy, enhancing the strength of cryptographic algorithms. Overall, the TRNG plays a crucial role in ensuring the security and robustness of various systems that rely on random number generation.

14.2 Features

- Has an internal source of entropy based on a chain of digital inverters.

  The inverter cells come from a standard cell library without requiring any specialized cells.

  – An odd number of inverters results in a continuous oscillation while active.
- Generates 10K bits per second of entropy when the core is running at 200 MHz.
- Includes built-in hardware tests for autocorrelation and Continuous Random Number Generation Testing (CRNGT) in compliance with the following standards:
  – *FIPS 140-2, Security Requirements for Cryptographic Modules*.
  – *AIS-31, Functionality Classes and Evaluation Methodology for True Random Number Generators*.
- Includes an AMBA APB2 slave interface.

14.3 Functional Description

14.3.1 Interface

Figure 14-1 shows a top view of the TRNG and its interfaces.

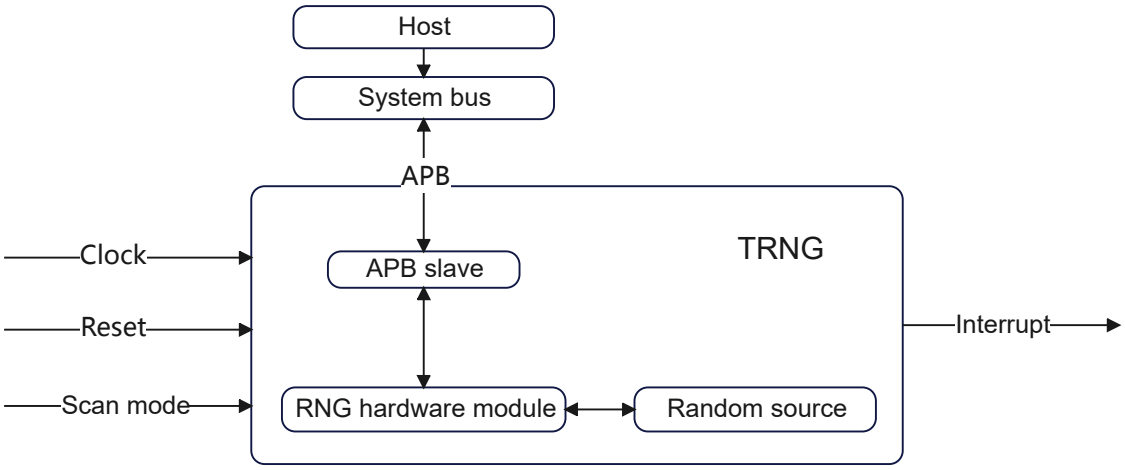


Figure 14-1 TRNG Hardware Overview

The TRNG has several interfaces:

- Clock interface: This is the input for the clock signal. The TRNG uses this clock signal to time its operations.

- Reset interface: This is used to reset the TRNG to its initial state. When a reset signal is received, the TRNG stops its current operation and starts a new one from scratch.
- APB interface: This is used for the communication of the TRNG with the processor. It allows the processor to read the random numbers generated by the TRNG, and also to control its operation.
- Interrupt: This is used to alert the processor when the TRNG has finished generating a random number. The TRNG sends an interrupt signal to the processor, which then reads the random number from the APB interface.

14.3.1.1 Clock Interface

The TRNG only uses rng_clk as its clock. There is no clock gating mechanism implemented within the TRNG.

This ensures that the TRNG consistently generates random numbers without interruptions or patterns. The absence of clock gating guarantees that the TRNG does not miss any random events and provides a high level of randomness for cryptographic applications.

14.3.1.2 Reset Interface

The reset interface consists of the sys_rst_n signal, which is the asynchronous global reset signal of the TRNG.

The sys_rst_n signal is essential for initializing the TRNG and ensuring its proper functioning. As an asynchronous global reset signal, sys_rst_n triggers the reset mechanism within the TRNG, allowing it to start in a known state.

The reset is generated by the SoC, granting it control over the TRNG's initialization process. This guarantees a proper reset of the TRNG, thereby ensuring its ability to generate random numbers reliably.

14.3.1.3 APB Slave Interface

The TRNG is connected to the SoC system bus as an APB slave. This interface facilitates access to the TRNG by a host processor.

Communication between the host processor and the TRNG occurs through reading and writing to specific registers on the APB slave interface. This functionality empowers the host processor to both request random numbers and configure the TRNG's operation.

14.3.1.4 Interrupt Interface

The cc_host_int_req (active-HIGH) functions as the TRNG's interrupt output to the SoC.

The cc_host_int_req output stays high until the SoC clears the interrupt source bits in the RNG_ICR register.

NOTE: You need to connect this interrupt to your SoC's interrupt controller.

14.4 Registers

14.4.1 Register Address Map

Offset	Register Name	Register Description
0x0100	TRNG_RNGIMR	TRNG interrupt masking register
0x0104	TRNG_RNGISR	TRNG interrupt and status register
0x0108	TRNG_RNGICR	TRNG interrupt clear register
0x010c	TRNG_TRNGCONFIG	TRNG configuration register
0x0110	TRNG_TRNGVALID	TRNG data valid register
0x0114	TRNG_EHRDATA0	TRNG entropy harvesting data register 0
0x0118	TRNG_EHRDATA1	TRNG entropy harvesting data register 1
0x011c	TRNG_EHRDATA2	TRNG entropy harvesting data register 2
0x0120	TRNG_EHRDATA3	TRNG entropy harvesting data register 3
0x0124	TRNG_EHRDATA4	TRNG entropy harvesting data register 4
0x0128	TRNG_EHRDATA5	TRNG entropy harvesting data register 5
0x012c	TRNG_RNDSOURCEENABLE	TRNG random source enable register
0x0130	TRNG_SAMPLECNT1	TRNG sample control register
0x0134	TRNG_AUTOCORRSTATISTIC	TRNG autocorrelation test register
0x0138	TRNG_TRNGDEBUGCONTROL	TRNG debug control register
0x0140	TRNG_TRNGSWRESET	TRNG software reset register
0x01b4	TRNG_RNGDEBUGENINPUT	TRNG debug enable register
0x01b8	TRNG_RNGBUSY	TRNG busy register

Offset	Register Name	Register Description
0x01bc	TRNG_RSTBITSCOUNTER	TRNG reset bits counter register
0x01c0	TRNG_RNGVERSION	TRNG version register
0x01e0	TRNG_RNGBISTCNTR0	TRNG built-in self-test counter register 0
0x01e4	TRNG_RNGBISTCNTR1	TRNG built-in self-test counter register 1
0x01e8	TRNG_RNGBISTCNTR2	TRNG built-in self-test counter register 2

14.4.2 Register Field Details

14.4.2.1 TRNG_RNGIMR

0x0100			TRNG interrupt masking register											TRNG_RNGIMR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												VNERRINTMASK	CRNGTERRINTMASK	AUTOCORRERRINTMASK	EHRVALIDINTMASK
Type	RO												RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 14-1 TRNG Interrupt Masking Register Description

Field	Name	Description
31:4	Reserved	Reserved
3	VNERRINTMASK	1: Masks the von Neumann error interrupt. No interrupt is generated.
2	CRNGTERRINTMASK	1: Masks the CRNGT error interrupt. No interrupt is generated.
1	AUTOCORRERRINTMASK	1: Masks the autocorrelation interrupt. No interrupt is generated.
0	EHRVALIDINTMASK	1: Masks the EHR interrupt. No interrupt is generated.

14.4.2.2 TRNG_RNGISR

0x0104			TRNG interrupt and status register											TRNG_RNGISR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												VNERR	CRNGTERR	AUTOCORRERR	EHRVALID
Type	RO												RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-2 TRNG Interrupt and Status Register Description

Field	Name	Description
31:4	Reserved	Reserved
3	VNERR	When set to 1, it indicates a von Neumann error. A von Neumann error occurs if 32 consecutive collected bits are identical, that is, 32 zeros or 32 ones.
2	CRNGTERR	When set to 1, it indicates a Continuous Random Number Generation Testing (CRNGT) error in the TRNG test failed. Failure occurs when two consecutive blocks of 16 collected bits are equal.
1	AUTOCORRERR	When set to 1, it indicates that the Autocorrelation test failed four times in a row. When set, the TRNG stops functioning until the next reset.

Field	Name	Description
0	EHRVALID	Set to 1 when 192 bits have been collected in the TRNG, and the EHR_DATA[0, 1, 2, 5] registers are ready to be read.

14.4.2.3 TRNG_RNGICR

0x0108			TRNG interrupt clear register											TRNG_RNGICR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												VNERR	CRNGTERR	AUTOCORRERR	EHRVALID
Type	RO												WC	WC	WC	WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-3 TRNG Interrupt Clear Register Description

Field	Name	Description
31:4	Reserved	Reserved
3	VNERR	Set to 1 to clear a von Neumann error.
2	CRNGTERR	Set to 1 to clear a Continuous Random Number Generation Testing (CRNGT) error.
1	AUTOCORRERR	Software cannot clear this bit. Only a TRNG reset can clear this bit.
0	EHRVALID	Set to 1 after the EHR_DATA[0,1,2, 5] registers have been read.

14.4.2.4 TRNG_TRNGCONFIG

0x010c			TRNG configuration register											TRNG_TRNGCONFIG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														RNDSRCSEL	
Type	RO														RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-4 TRNG Configuration Register Description

Field	Name	Description
31:2	Reserved	Reserved
1:0	RNDSRCSEL	Defines the length of the oscillator ring (the number of inverters) out of four possible selections.

14.4.2.5 TRNG_TRNGVALID

0x0110			TRNG data valid register											TRNG_TRNGVALID		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															EHRVALID
Type	RO															RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-5 TRNG Data Valid Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	EHRVALID	1'b1 indicates that the collection of bits in the TRNG is completed, and data can be read from the EHR_DATA registers.

14.4.2.6 TRNG_EHRDATA0

0x0114			TRNG entropy harvesting data register 0											TRNG_EHRDATA0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-6 TRNG Entropy Harvesting Data Register 0 Description

Field	Name	Description
31:0	EHRDATA	Return bits [31:0] of the EHR

14.4.2.7 TRNG_EHRDATA1

0x0118			TRNG entropy harvesting data register 1											TRNG_EHRDATA1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-7 TRNG Entropy Harvesting Data Register 1 Description

Field	Name	Description
31:0	EHRDATA	Return bits [63:32] of the EHR

14.4.2.8 TRNG_EHRDATA2

0x011c			TRNG entropy harvesting data register 2											TRNG_EHRDATA2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-8 TRNG Entropy Harvesting Data Register 2 Description

Field	Name	Description
31:0	EHRDATA	Return bits [95:64] of the EHR

14.4.2.9 TRNG_EHRDATA3

0x0120			TRNG entropy harvesting data register 3											TRNG_EHRDATA3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-9 TRNG Entropy Harvesting Data Register 3 Description

Field	Name	Description
31:0	EHRDATA	Return bits [127:96] of the EHR

14.4.2.10 TRNG_EHRDATA4

0x0124			TRNG entropy harvesting data register 4											TRNG_EHRDATA4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-10 TRNG Entropy Harvesting Data Register 4 Description

Field	Name	Description
31:0	EHRDATA	Return bits [159:128] of the EHR

14.4.2.11 TRNG_EHRDATA5

0x0128			TRNG entropy harvesting data register 5											TRNG_EHRDATA5		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EHRDATA															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-11 TRNG Entropy Harvesting Data Register 5 Description

Field	Name	Description
31:0	EHRDATA	Return bits [191:160] of the EHR

14.4.2.12 TRNG_RNDSOURCEENABLE

0x012c			TRNG random source enable register											TRNG_RNDSOURCEENABLE		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															RNDSRCEN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-12 TRNG Random Source Enable Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	RNDSRCEN	The enable signal for the random source.

14.4.2.13 TRNG_SAMPLECNT1

0x0130			TRNG sample control register											TRNG_SAMPLECNT1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	SAMPLECNTR1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SAMPLECNTR1															
Type	RW															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 14-13 TRNG Sample Control Register Description

Field	Name	Description
31:0	SAMPLECNTR1	The number of rng_clk cycles between two consecutive ring oscillator samples.

14.4.2.14 TRNG_AUTOCORRSTATISTIC

0x0134			TRNG autocorrelation test register											TRNG_AUTOCORRSTATISTIC		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										AUTOCORRFAILS					
Type	RO										RW					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	AUTOCORRFAILS		AUTOCORRTRYS													
Type	RW		RW													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-14 TRNG Autocorrelation Test Register Description

Field	Name	Description
31:22	Reserved	Reserved
21:14	AUTOCORRFAILS	Count each time an autocorrelation test fails. Any write to the register resets the counter. Stop collecting statistics if one of the counters has reached the limit.
13:0	AUTOCORRTRYS	Counts each time an autocorrelation test starts. Any write to the register resets the counter. Stop collecting statistics if one of the counters has reached the limit.

14.4.2.15 TRNG_TRNGDEBUGCONTROL

0x0138			TRNG debug control register											TRNG_TRNGDEBUGCONTROL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												AUTOCORRELATEBYPASS	TRNGCRNGTBYPASS	VNCBYPASS	Reserved
Type	RO												RW	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-15 TRNG Debug Control Register Description

Field	Name	Description
31:4	Reserved	Reserved
3	AUTOCORRELATEBYPASS	When this bit is set, the autocorrelation test in the TRNG module is bypassed.
2	TRNGCRNGTBYPASS	When this bit is set, the CRNGT test in the TRNG is bypassed.
1	VNCBYPASS	When this bit is set, the von Neumann balancer is bypassed (including the 32 consecutive bits test).
0	Reserved	Reserved

14.4.2.16 TRNG_TRNGSWRESET

0x0140			TRNG software reset register											TRNG_TRNGSWRESET		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															RNGSWRESET
Type	RO															WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-16 TRNG Software Reset Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	RNGSWRESET	Any value written (1'b0 or 1'b1) causes a reset cycle to the TRNG block. The reset mechanism takes about four RNG clock cycles until the reset line is deasserted.

14.4.2.17 TRNG_RNGDEBUGENINPUT

0x01b4			TRNG debug enable register											TRNG_RNGDEBUGENINPUT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															RNGDEBUGEN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-17 TRNG Debug Enable Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	RNGDEBUGEN	Reflect the rng_debug_enable input port

14.4.2.18 TRNG_RNGBUSY

0x01b8			TRNG busy register											TRNG_RNGBUSY		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															RNGBUSY
Type	RO															RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-18 TRNG Busy Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	RNGBUSY	Reflect the status of the rng_busy signal

14.4.2.19 TRNG_RSTBITSCOUNTER

0x01bc			TRNG reset bits counter register											TRNG_RSTBITSCOUNTER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															RSTBITSCOUNTER
Type	RO															WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-19 TRNG Reset Bits Counter Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	RSTBITSCOUNTER	Writing any value to this bit resets the bits counter and TRNG_VALID registers. The TRNG_RNDSOURCEENABLE register must be unset for the reset to take place.

14.4.2.20 TRNG_RNGVERSION

0x01c0			TRNG version register											TRNG_RNGVERSION		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								RNGUSE5SBOXES	RESEEDINGEXISTS	KATEXISTS	PRNGEXISTS	TRNGTESTSBYPASSEN	AUTOCORREXISTS	CRNGTEXISTS	EHRWIDTH192
Type	RO								RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Table 14-20 TRNG Version Register Description

Field	Name	Description
31:8	Reserved	Reserved
7	RNGUSE5SBOXES	1'b0: 20 SBOX AES 1'b1: 5 SBOX AES
6	RESEEDINGEXISTS	1'b0: Does not exist 1'b1: Exists
5	KATEXISTS	1'b0: Does not exist 1'b1: Exists
4	PRNGEXISTS	1'b0: Does not exist 1'b1: Exists
3	TRNGTESTSBYPASSEN	1'b0: TRNG tests bypass not enabled

Field	Name	Description
		1'b1: TRNG tests bypass enabled
2	AUTOCORREXISTS	1'b0: Does not exist 1'b1: Exists
1	CRNGTEXISTS	1'b0: Does not exist 1'b1: Exists

14.4.2.21 TRNG_RNGBISTCNTR0

0x01e0			TRNG built-in self-test counter register 0											TRNG_RNGBISTCNTR0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										ROSCCNTRVAL					
Type	RO										RO					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ROSCCNTRVAL															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-21 TRNG Built-in Self-Test Counter Register 0 Description

Field	Name	Description
31:22	Reserved	Reserved
21:0	ROSCCNTRVAL	Return the results of the TRNG BIST counter

14.4.2.22 TRNG_RNGBISTCNTR1

0x01e4			TRNG built-in self-test counter register 1											TRNG_RNGBISTCNTR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										ROSCCNTRVAL					
Type	RO										RO					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ROSCCNTRVAL															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-22 TRNG Built-in Self-Test Counter Register 1 Description

Field	Name	Description
31:22	Reserved	Reserved
21:0	ROSCCNTRVAL	Return the results of the TRNG BIST counter

14.4.2.23 TRNG_RNGBISTCNTR2

0x01e8			TRNG built-in self-test counter register 2											TRNG_RNGBISTCNTR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										ROSCCNTRVAL					
Type	RO										RO					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ROSCCNTRVAL															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-23 TRNG Built-in Self-Test Counter Register 2 Description

Field	Name	Description
31:22	Reserved	Reserved
21:0	ROSCCNTRVAL	Return the results of the TRNG BIST counter

Advanced Encryption Standard (AES)

This chapter describes the details of the Advanced Encryption Standard (AES) module.

Topics: **Page**

15.1 Introduction

The Advanced Encryption Standard (AES) hardware accelerator is designed to perform encryption or decryption of data using an algorithm and implementation that adhere completely to the AES specifications outlined in Federal Information Processing Standards (FIPS) publication 197.

AES supports various chaining modes such as CTR, OFB, CFB, ECB, and CBC and can handle key sizes of 128, 192, or 256 bits. The AES accelerator is an AMBA AHB slave peripheral that can be accessed only through 32-bit single accesses. Attempting other access types may result in an AHB error and can potentially corrupt the register content. Additionally, the peripheral supports DMA single transfers for both incoming and outgoing data, requiring two DMA channels.

15.2 Features

- Compliance with NIST “Advanced encryption standard (AES), FIPS publication 197” from November 2001
- 128-bit data block processing
- Support for cipher key lengths of 128-bit,192-bit and 256-bit
- Encryption and decryption with multiple chaining modes:
  - Electronic Codebook (ECB) mode
  - Cipher Block Chaining (CBC) mode
  - Counter (CTR) mode
  - Cipher Feedback (CFB) mode
  - Output Feedback (OFB) mode
- 20 AHB clock cycle latency in ECB mode for processing one 128-bit block of data with, respectively, 128-bit, 192-bit or 256-bit key
- Integrated round key scheduler to compute the last round key for ECB/CBC/CTR/CFB/OFB decryption
- AHB slave peripheral, accessible through 32-bit word single accesses only
- 256-bit register for storing the cryptographic key (eight 32-bit registers)
- 128-bit register for storing initialization vector (four 32-bit registers)
- 4x32-bit buffer for data input and output
- Automatic data flow control with support of single-transfer Direct Memory Access (DMA) using two channels (one for incoming data, one for processed data)
- Data-swapping logic to support 1-bit, 8-bit, 16-bit or 32-bit data

15.3 Functional Description

15.3.1 Block Diagram

Figure 15-1 shows the block diagram of the AES module.

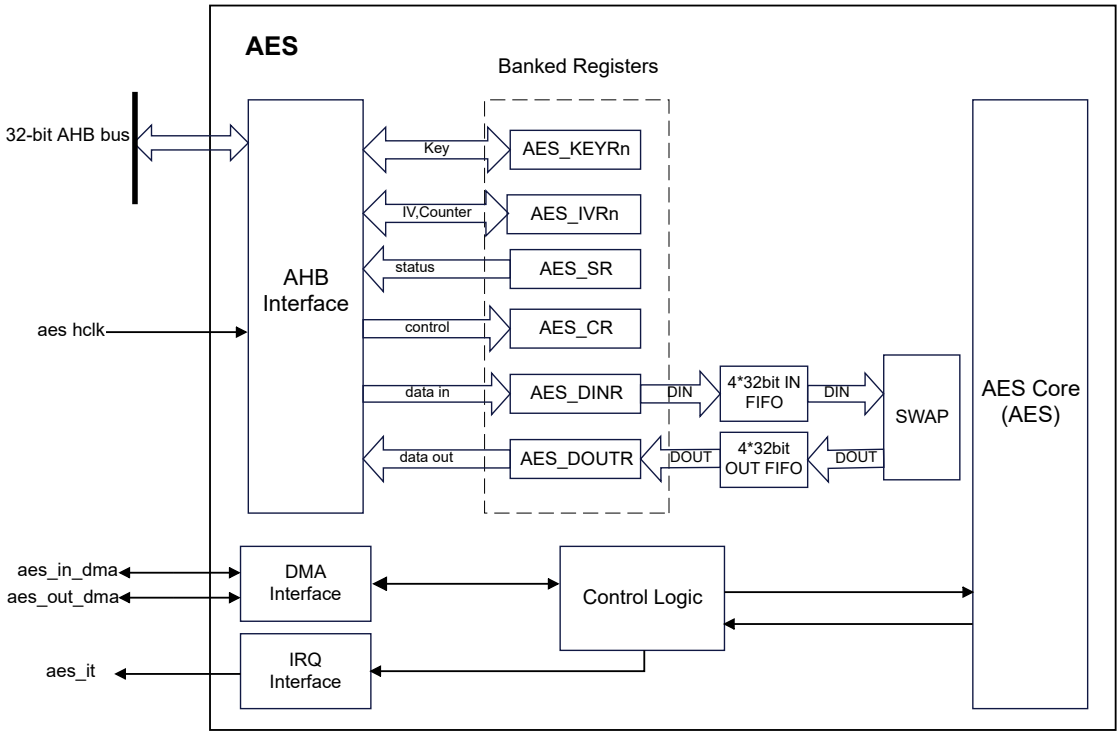


Figure 15-1 AES Block Diagram

15.3.2 Cryptographic Core

15.3.2.1 Overview

The AES cryptographic core comprises the following components:

- AES core algorithm (AEA)
- Key input
- Initialization Vector (IV) input
- Chaining algorithm logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks (four words) with 128-bit, 192-bit or 256-bit key length. Depending on the chaining mode, the AES requires zero or one 128-bit IV. The AES features the following modes of operation:

- Mode 1: Plaintext encryption using a key stored in the AES_KEYRx registers
- Mode 2: Ciphertext decryption using a key stored in the AES_KEYRx registers. The operating mode is selected by programming the MODE bitfield of the AES_CR register. It may be done only when the AES peripheral is disabled.

15.3.2.2 Typical Data Processing

Typical usage of the AES is described in Encryption Process.

NOTE: The outputs of the intermediate AEA stages are never revealed outside the cryptographic boundary, with the exclusion of the IVI bitfield.

The intermediate AEA stages are always kept confidential within the cryptographic boundary, except for the IVI bitfield.

15.3.2.3 Chaining Modes

AES supports different chaining modes, which can be selected using the CHMOD[2:0] bit field in the AES_CR register. These modes include Electronic Code Book (ECB), Cipher Block Chaining (CBC), Counter (CTR), Cipher Feedback (CFB), and Output Feedback (OFB) modes.

NOTE: The chaining mode can only be changed when the AES is disabled. This is indicated by the EN bit in the AES_CR Register being set to 0.

The following subsections explain the principle of each AES chaining mode. More detailed information about each mode can be found in dedicated sections, starting from Chaining Modes.

15.3.2.3.1 Electronic Codebook (ECB) Mode

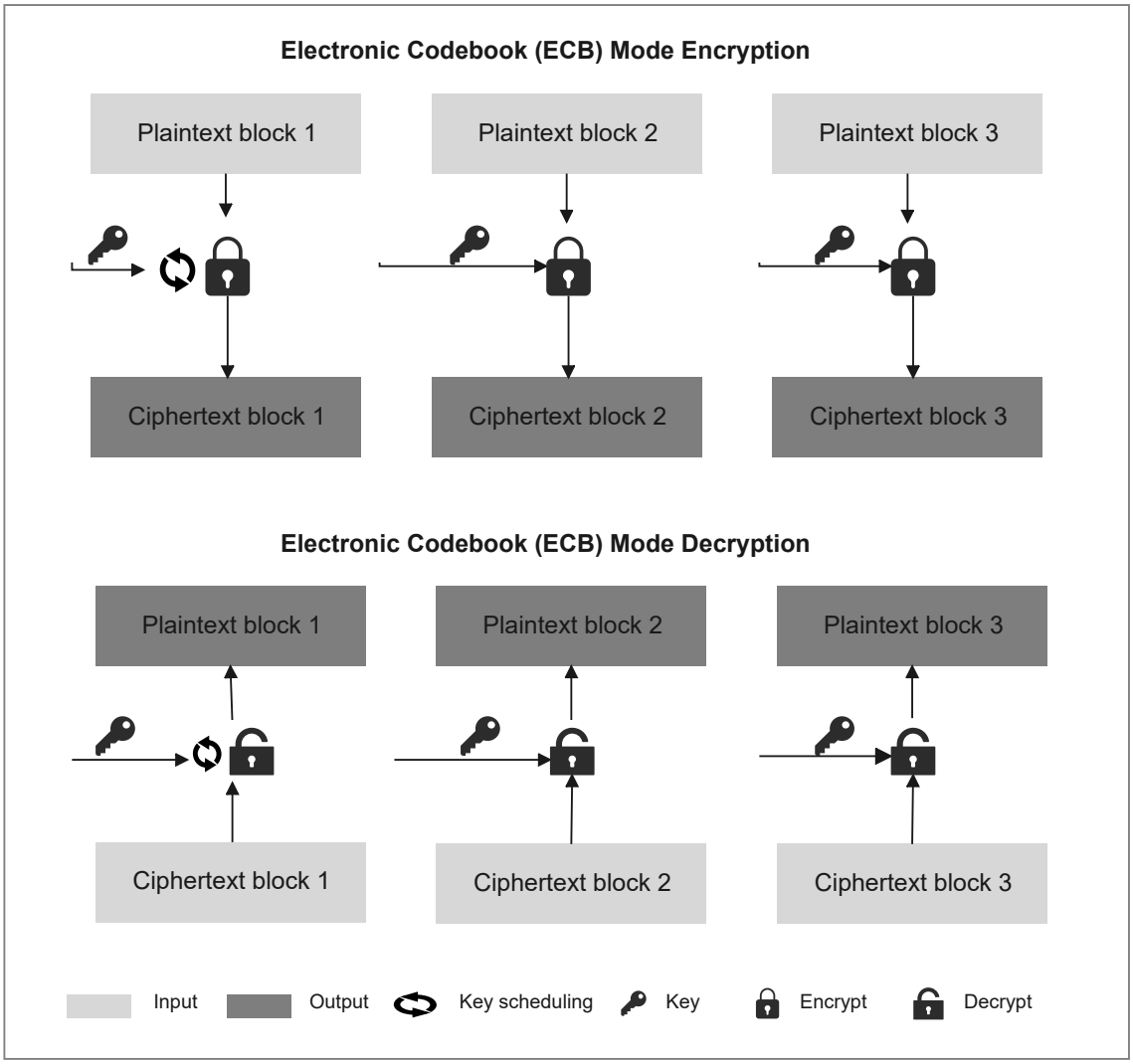


Figure 15-2 ECB Encryption and Decryption Principle

ECB is the simplest mode of operation. It does not involve any chaining operations or special initialization stage. The message is divided into blocks, each encrypted or decrypted independently.

NOTE: For decryption, a specific key scheduling is necessary prior to processing the initial block.

15.3.2.3.2 Cipher Block Chaining (CBC) Mode

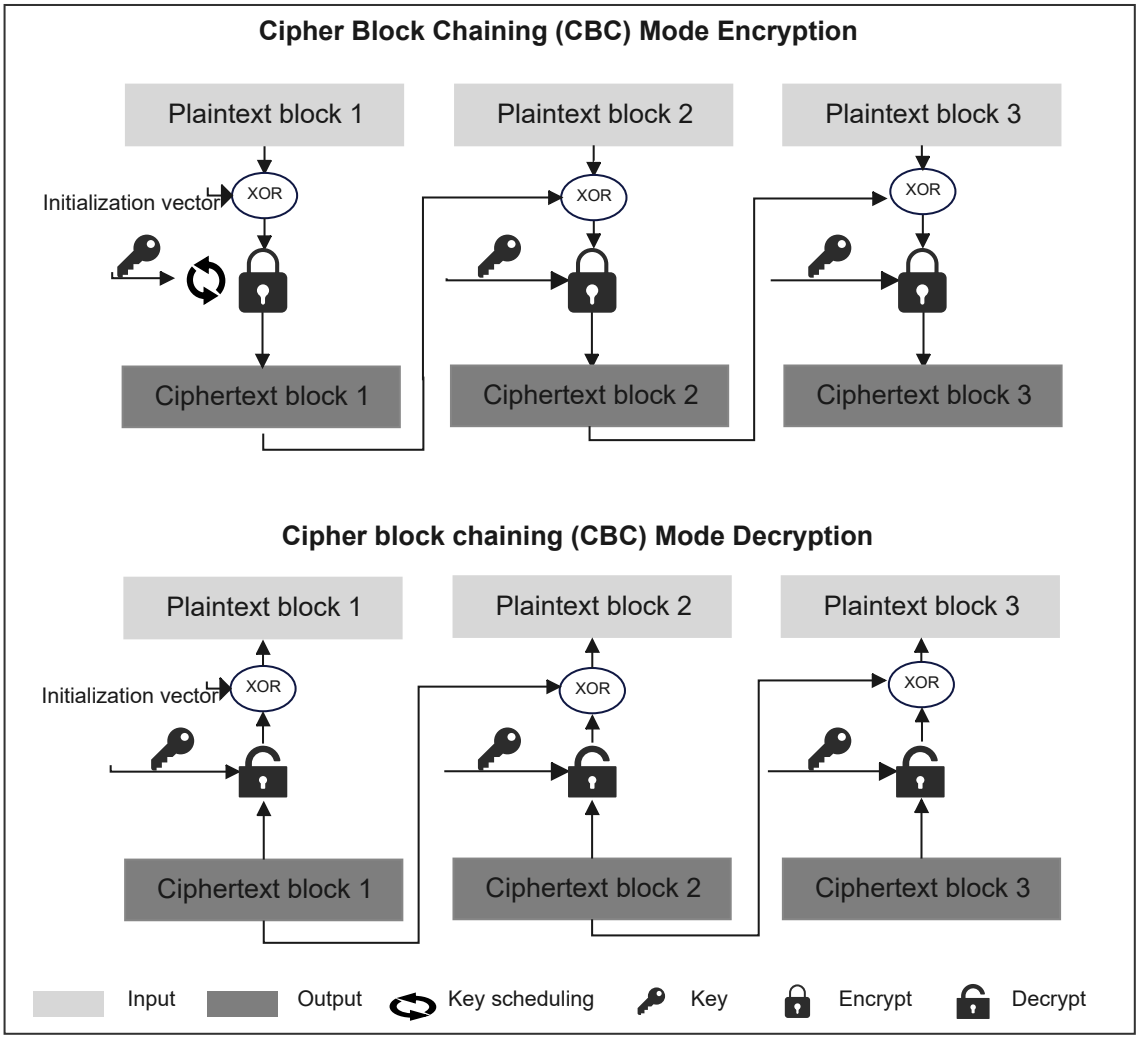


Figure 15-3 CBC Encryption and Decryption Principle

Each block in CBC mode is linked to the input of the next block. To ensure the uniqueness of each message, an initialization vector is used during the processing of the first block.

NOTE: Prior to decrypting the initial block, it is essential to perform special key scheduling.

15.3.2.3.3 Counter (CTR) Mode

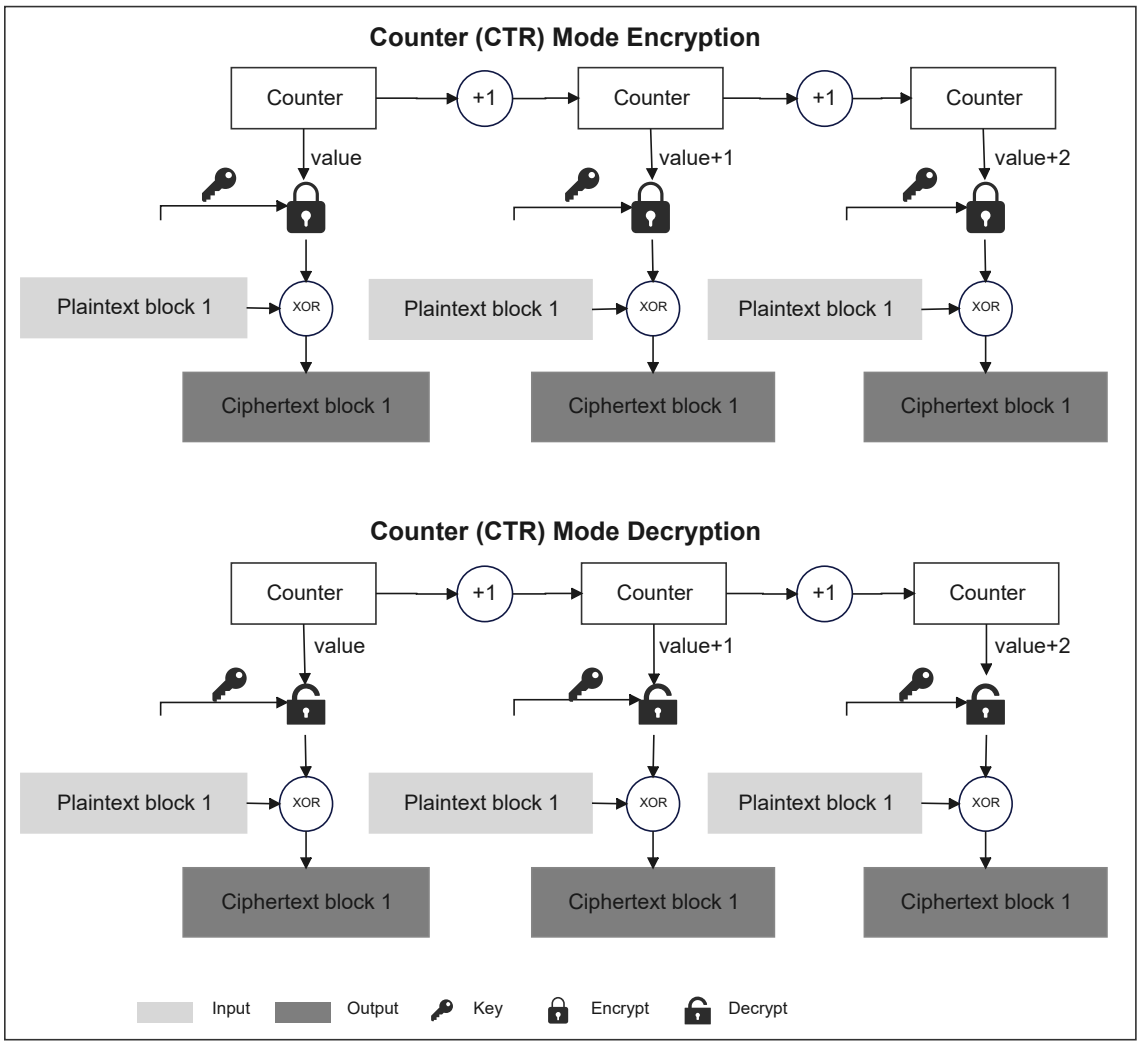


Figure 15-4 CTR Encryption and Decryption Principle

The CTR mode uses the AES core to generate a key stream, which is then XOR-ed with the plaintext to derive the ciphertext, as outlined in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.

NOTE: Unlike with ECB and CBC modes, no key scheduling is necessary for CTR decryption. In this chaining scheme, the AES core is consistently used in encryption mode to generate the key stream or counter blocks.

15.3.2.3.4 Cipher Feedback (CFB) Mode

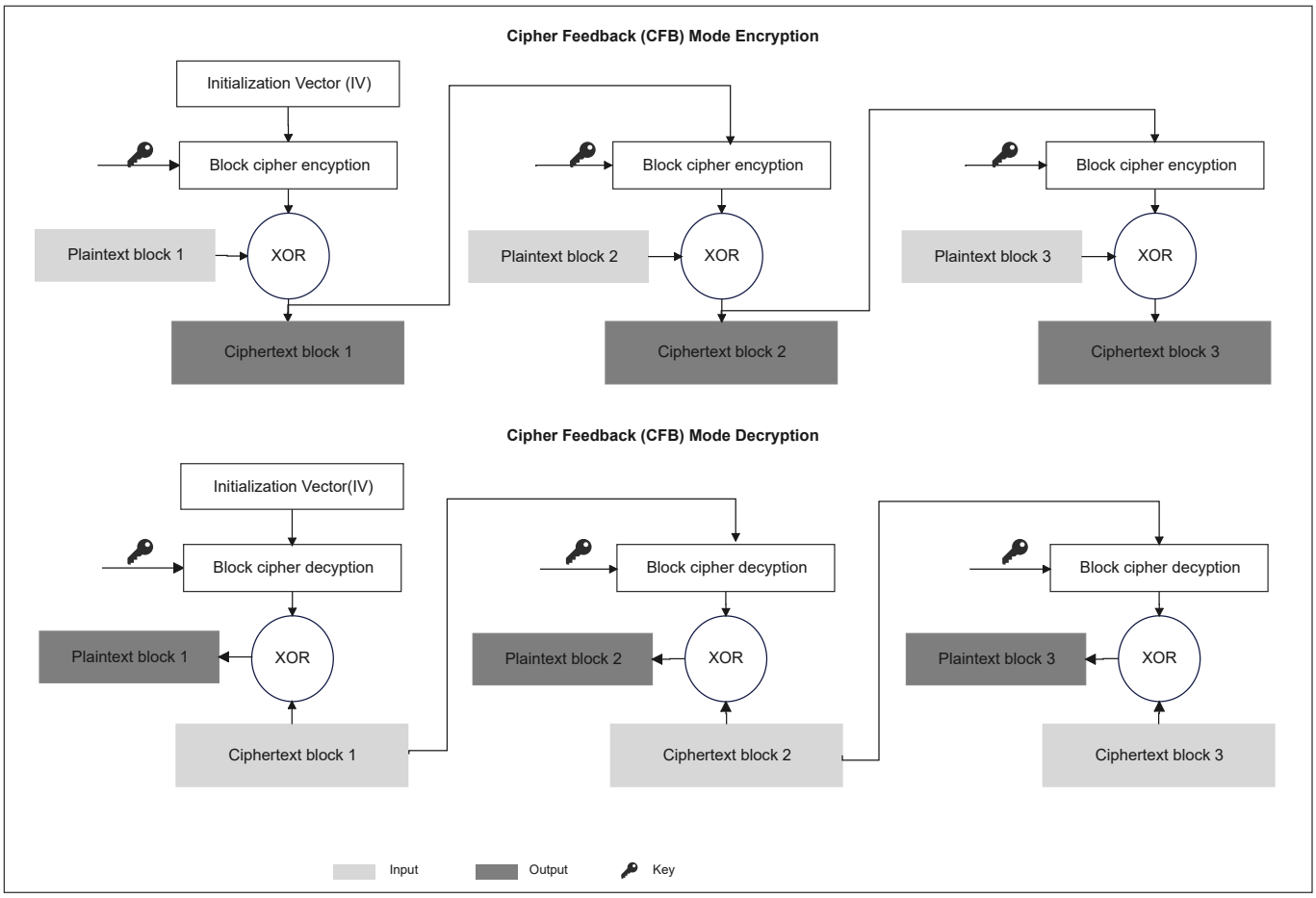


Figure 15-5 CFB Encryption and Authentication Principle

In this mode, the Initialization Vector (IV) is used, which can be a random block of text. The first block of plaintext and IV are combined using the XOR operation and then encrypt the resultant message using the key and form the first block of ciphertext. The first block of ciphertext is used as the IV for the second block of plaintext. The same procedure is followed for all blocks of plaintext.

When received, the ciphertext is divided into blocks. The first ciphertext block is decrypted using the same key used for encryption. The decrypted result is then XORed with the IV to form the first plaintext block. The second block of ciphertext is also decrypted using the same key, and the decrypted result is XORed with the first block of ciphertext to form the second block of plaintext. This procedure is repeated for all blocks.

15.3.2.3.5 Output Feedback (OFB) Mode

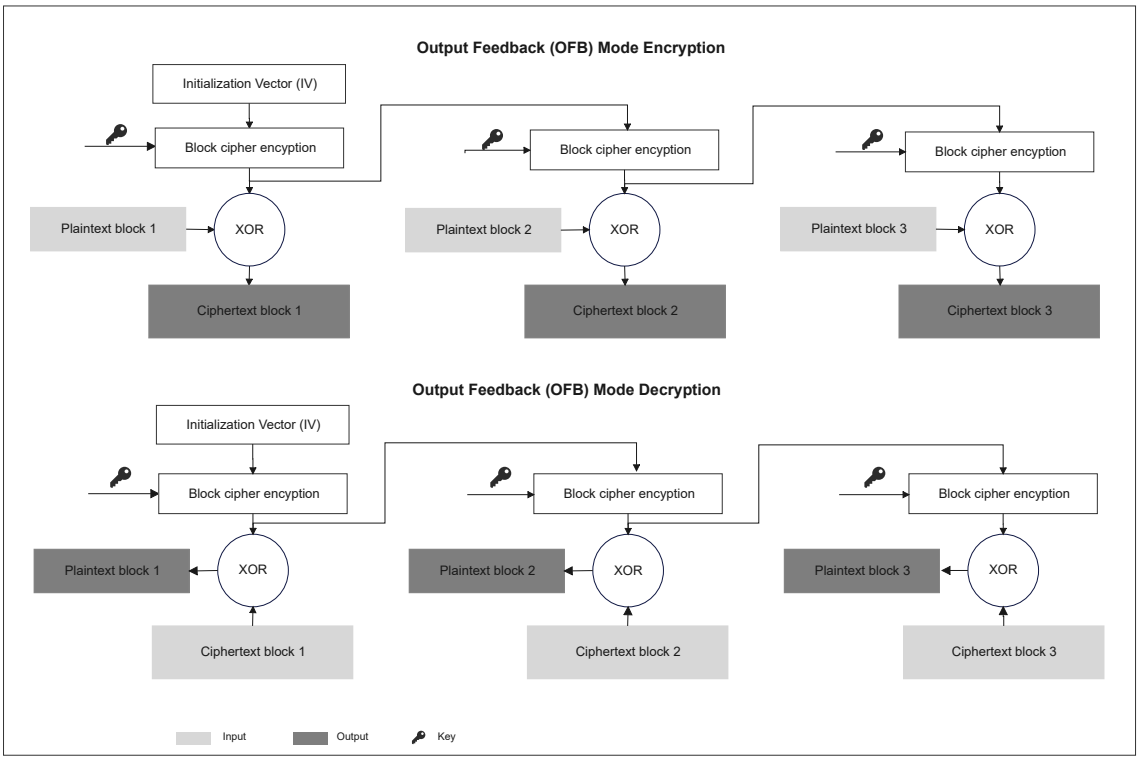


Figure 15-6 OFB Encryption and Authentication Principle

Output Feedback (OFB) mode is a mode of operation in encryption. It is similar to CFB mode, with the main difference being that in CFB mode, the ciphertext is used for the next stage of encryption, whereas in OFB mode, the output of the IV encryption is used for the next stage.

The IV is encrypted using the key and becomes the encrypted IV. The plaintext and encrypted IV are then combined using XOR to produce the ciphertext. In the next stage, the ciphertext from the previous stage is used as the IV for the next iteration. The same procedure is followed for all blocks.

15.3.3 Encryption Process

A typical cipher operation is explained below. Detailed information is provided in sections starting from Chaining Mode.

15.3.3.1 Initialization

To initialize AES, disable it by clearing the EN bit of the AES_CR register. Then, perform the following steps in any order:

- Configure the AES mode by programming the MODE bitfield of the AES_CR register.
  - For encryption, select Mode 1 (MODE = 0).
  - For decryption, select Mode 2 (MODE = 1).
- Select the chaining mode by programming the CHMOD[2:0] bit field of the AES_CR register.
- Configure the data type (1-bit, 8-bit, 16-bit or 32-bit), with the DATATYPE[1:0] bit field in the AES_CR register.
- When it is required (for example, in CBC or CTR chaining modes), write the initialization vector into the AES_IVRx registers.
- Configure the key size (128-bit, 192-bit or 256-bit), with the KEYSIZE bitfield of the AES_CR register.
- Write a symmetric key into the AES_KEYRx registers (4 or 8 registers depending on the key size).

15.3.3.2 Data Append

This section describes different ways of appending data for processing, where the size of data to process is not a multiple of 128 bits

For ECB or CBC mode, refer to Ciphertext Stealing and Data Padding. The last block management in these cases is more complex than in the sequence described in this section.

Data Append through Polling

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral by setting the EN bit of the AES_CR register.
2. Repeat the following sub-sequence until the payload is entirely processed:
   a. Write four input data words into the AES_DINR register.
   b. Wait for the CCF status flag to be set in the AES_SR, then read the four data words from the AES_DOUTR register.
   c. Clear the CCF flag by setting the CCFC bit of the AES_CR register.
   d. If the data block that was just processed is the second-to-last block of the message and the remaining data in the last block to be processed is less than 128 bits, pad the rest of the last block with zeros, but make sure to set NPBLB.
3. As it is the last block, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES_CR register.

Data Append Using Interrupt

The method uses an interrupt from the AES peripheral to control the data append through the following sequence:

1. Enable interrupts from AES by setting the CCFIE bit of the AES_CR register.
2. Enable the AES peripheral by setting the EN bit of the AES_CR register.
3. Write the first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine upon interrupt:
   a. Read four output data words from the AES_DOUTR register.
   b. Clear the CCF flag and thus the pending interrupt by setting the CCFC bit of the AES_CR register.
   c. If the data block just processed is the second-last block of a message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros.
   d. If the data block just processed is the last block of the message, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES_CR register and quit the interrupt service routine.
   e. Write the next four input data words into the AES_DINR register and quit the interrupt service routine.

NOTE: AES is tolerant of delays between consecutive read or write operations, allowing interruptions from other peripherals, such as an interrupt from another peripheral, to be addressed between two AES computations.

Data Append Using DMA

With this method, all the transfers and processing are managed by DMA and AES. To use the method, proceed as follows:

1. Prepare the last four-word data block (if the data to process does not fill it completely), by padding the remainder of the block with zeros.
2. Configure the DMA controller to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in DMA Interface. Configure the DMA controller to generate an interrupt on transfer completion.
3. Enable the AES peripheral by setting the EN bit of the AES_CR register.
4. Enable DMA requests by setting the DMAINEN and DMAOUTEN bits of the AES_CR register.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

NOTE: The CCF flag is not applicable in this method. The AES_DOUTR register is automatically read by DMA at the end of the computation phase, without requiring any software action.

15.3.4 Decryption Round Key Preparation

Internal key schedule is used to generate AES round keys. In AES encryption, the round 0 key is the one stored in the key registers. AES decryption must start using the last round key. As the encryption key is stored in memory, a special key scheduling must be performed to obtain the decryption key. This key scheduling is only required for AES decryption in ECB and CBC modes.

The recommended method is to select Mode 0 by setting to 0 the MODE bitfield of the AES_CR (key process only), then proceed with the decryption by setting MODE to 1. Mode 1 usage is described below:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select Mode 0 by setting the MODE bitfield in the AES_CR to 0.
3. Specify the desired chaining mode using the CHMOD[2:0] bit field in the AES_CR register.
4. Set the key length to either 128, 192, or 256 bits by using the KEYSIZE bit in the AES_CR register.
5. Write the AES_KEYRx registers (128,192 or 256 bits) with an encryption key, as shown in Figure 15-7. Writes to the AES_IVRx registers have no effect.
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.
7. Wait for the CCF flag to be set in the AES_SR register. Once set, clear the CCF flag. The derived key is now available in the AES core and can be used for decryption.

NOTE: The AES is disabled by the hardware once the derived key is available. If you require faster speed, it is recommended to use DMA's Burst Mode, which allows for the transfer of 4 words at a time. Additionally, it is important to carefully review the relevant DMA configuration for optimal performance.

To restart a derivation key computation, repeat steps 4, 5, 6, and 7.

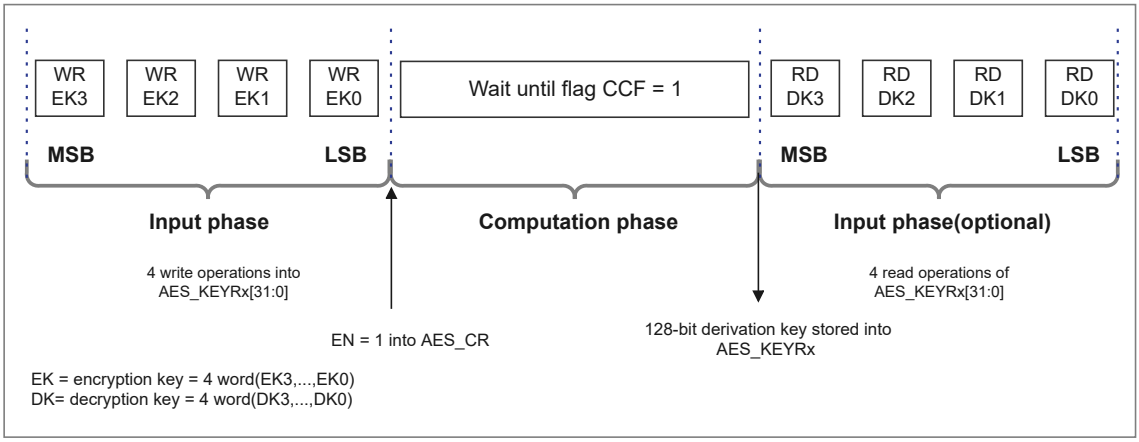


Figure 15-7 Encryption Key Derivation for ECB/CBC Decryption

If the software stores the initial key prepared for decryption, it is enough to do the key schedule operation only once for all the data to be decrypted with a given cypher key.

NOTE: The operation of the key preparation lasts 19 21 or 23 clock cycles, depending on the key size (128-bit, 192-bit or 256-bit), use DMA's Burst Mode, transfer 4 words at a time.

15.3.5 Ciphertext Stealing and Data Padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (128 bits), ciphertext stealing techniques are used, such as those described in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode. Since the AES peripheral does not support such techniques, the application must complete the last block of input data using data from the second last block.

NOTE: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, when AES is used in other modes than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 128 bits) must be padded with zeros prior to encryption (that is, extra bits must be appended to the trailing end of the data string). After decryption, the extra bits must be discarded. As AES does not implement automatic data padding operation to the last block, the application must follow the recommendation given in Encryption Process to manage messages the size of which is not a multiple of 128 bits.

NOTE: The swapping of padding data is done in the same manner as normal data, based on the DATATYPE[1:0] field of the AES_CR register. Refer to Data Registers and Data Swapping section for more information.

15.3.6 Chaining Modes

15.3.6.1 Overview

This section provides a concise explanation of the operation modes provided by the AES core. For more detailed information, please refer to the FIPS publication 197 dated November 26, 2001.

15.3.6.1.1 ECB Encryption and Decryption

ECB Encryption

Figure 15-8 illustrates the Electronic Codebook (ECB) encryption.

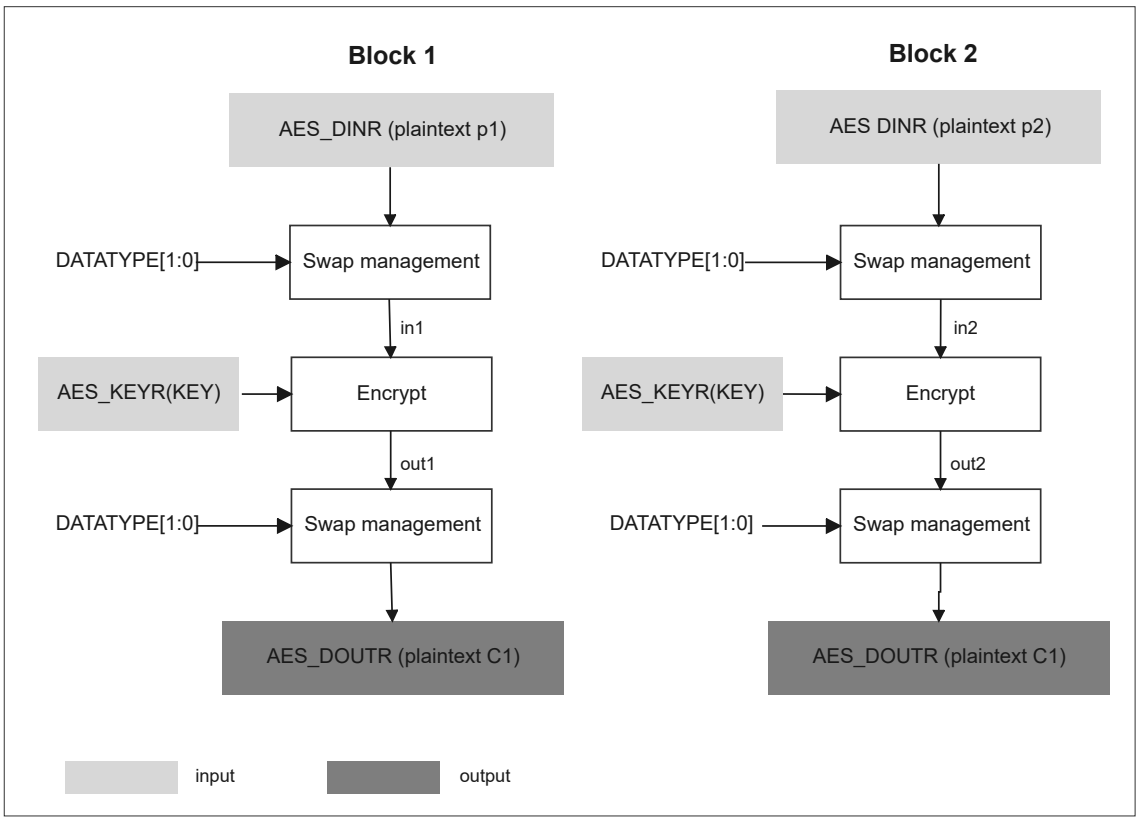


Figure 15-8 ECB Encryption

In ECB encrypt mode, the 128-bit plaintext input data block Px in the AES_DINR register first goes through bit/byte/half-word swapping. The swap result Inx is processed with the AES core set in encrypt mode, using a 128-bit, 192-bit or 256-bit key. The encryption result Outx goes through bit/byte/half-word swapping, then is stored in the AES_DOUTR register as 128-bit ciphertext output data block Cx. The ECB encryption continues in this way until the last complete plaintext block is encrypted.

ECB Decryption

Figure 15-9 illustrates the Electronic Codebook (ECB) decryption.

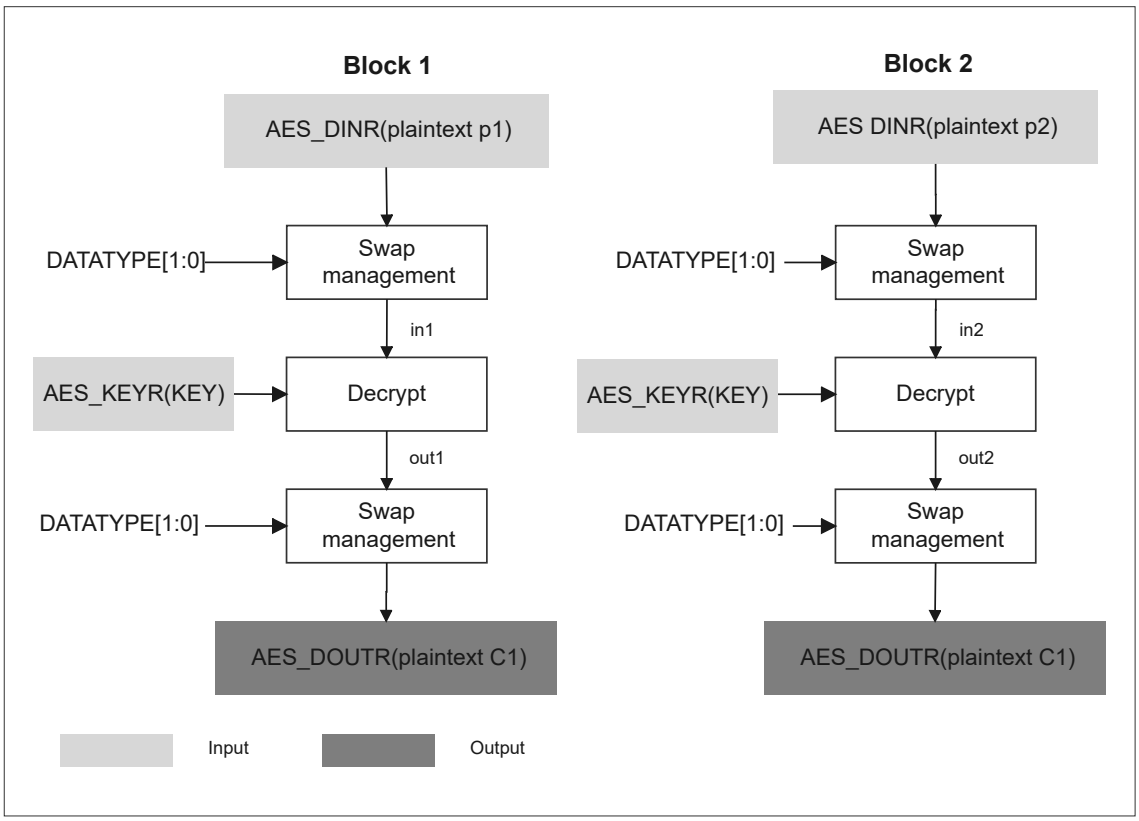


Figure 15-9 ECB Decryption

To perform an AES decryption in the ECB mode, the key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported by the AES core.

In ECB decrypt mode, the 128-bit ciphertext input data block C1 in the AES_DINR register first goes through bit/byte/half-word swapping. The keying sequence is reversed compared to that of the ECB encryption. The swap result I1 is processed with the AES core set in decrypt mode, using the formerly prepared decryption key. The decryption result goes through bit/byte/half-word swapping, then is stored in the AES_DOUTR register as 128-bit plaintext output data block P1. The ECB decryption continues in this way until the last complete ciphertext block is decrypted.

15.3.6.1.2 CBC Encryption and Decryption

CBC Encryption

Figure 15-10 illustrates the Cipher Block Chaining (CBC) encryption.

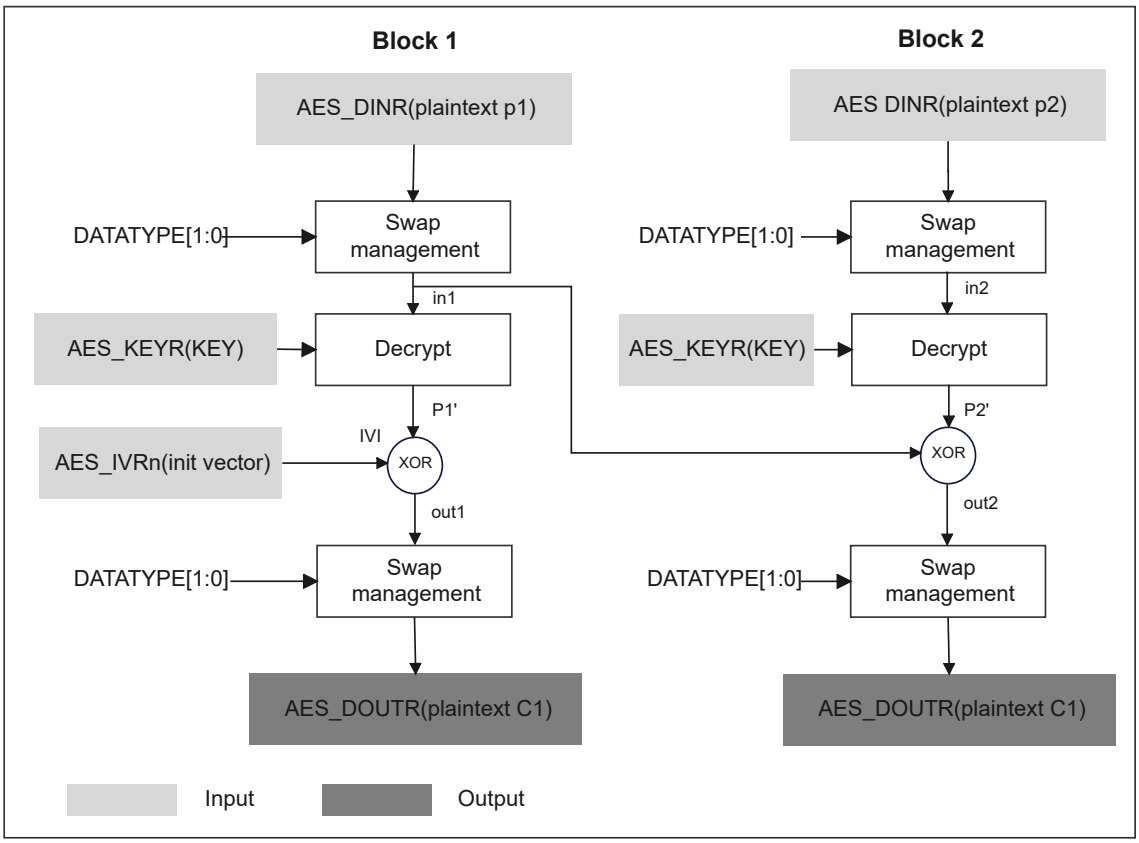


Figure 15-10 CBC Encryption

In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the In1 input data for encrypt with the AES core, using a 128-bit,192-bit or 256-bit key. The resulting 128-bit output block Out1, after swapping operation, is used as ciphertext C1. The Out1 data is then XOR-ed with the second-block plaintext data P2' to produce the I2 input data for the AES core to produce the second block of ciphertext data. The chaining of data blocks continues in this way until the last plaintext block in the message is encrypted.

If the message size is not a multiple of 128 bits, the final partial data block is encrypted in the way explained in Ciphertext Stealing and Data Padding.

CBC Decryption

Figure 15-11 illustrates the Cipher Block Chaining (CBC) decryption.

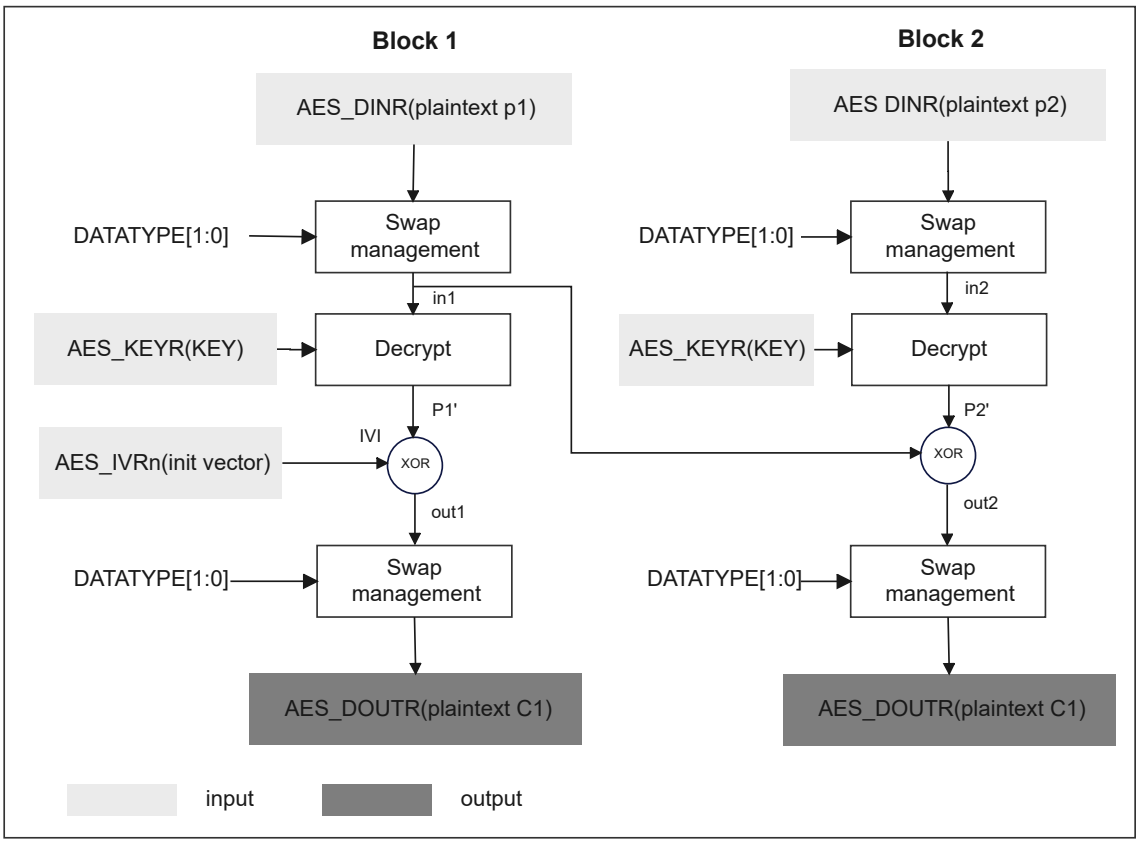


Figure 15-11 CBC Decryption

In CBC decrypt mode, like in ECB decrypt mode, the key must be prepared to perform an AES decryption.

After the key preparation process, the decryption goes as follows: the first 128-bit ciphertext block (after the swap operation) is used directly as the AES core input block In1 for decrypt operation, using the 128-bit, 192-bit or 256-bit key. Its output Out1 is XOR-ed with the 128-bit IVI field (that must be identical to that used during encryption) to produce the first plaintext block P1.

The second ciphertext block is processed in the same way as the first block, except that the In1 data from the first block is used in place of the initialization vector.

The decryption continues in this way until the last complete ciphertext block is decrypted.

If the message size is not a multiple of 128 bits, the final partial data block is decrypted in the way explained in Ciphertext Stealing and Data Padding.

For more information on data swapping, refer to Data Registers and Data Swapping.

15.3.6.1.3 CFB Encryption and Decryption

CFB Encryption

Figure 15-12 illustrates the Cipher Feedback (CFB) encryption.

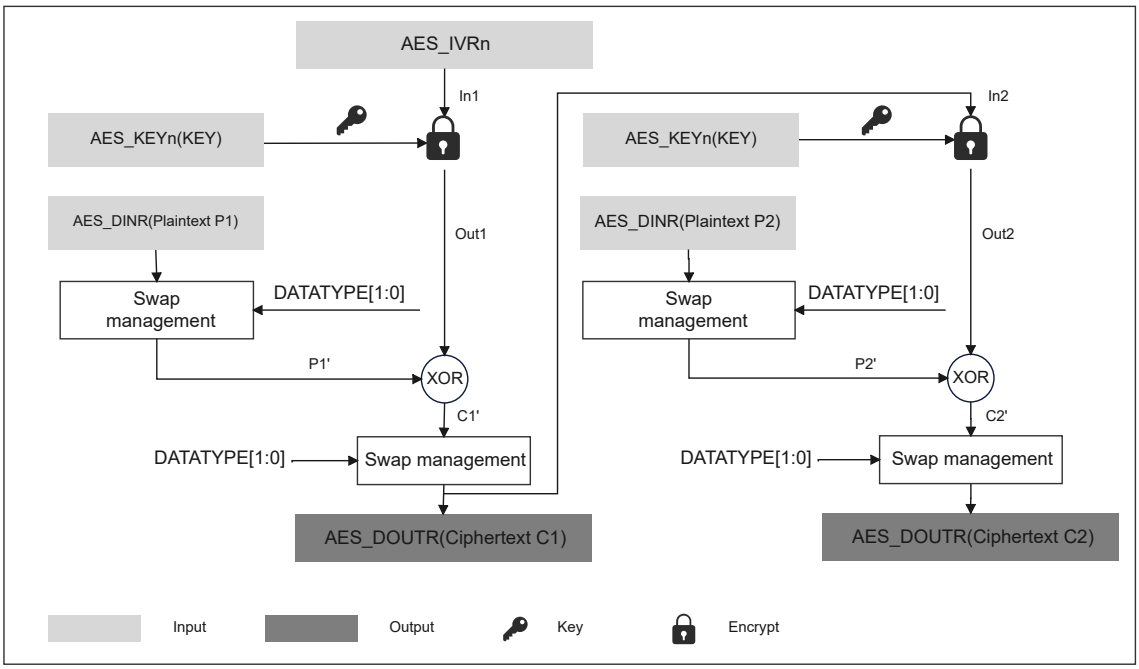


Figure 15-12 CFB Encryption

In CFB encrypt mode, the first plaintext input block produces the P1' after bit/byte/half-word swapping modules. a 128-bit IVI bitfield (initialization vector and counter) and a 128-bit,192-bit or 256-bit key produces Out1. P1' is XOR-ed with Out1, produce C1', which after bit/byte/half-word swapping as ciphertext C1.

The second ciphertext block is processed in the same way as the first block, except that the C1 data from the first block is used in place of the initialization vector.

The chaining of data blocks continues in this way until the last plaintext block in the message is encrypted.

CFB Decryption

Figure 15-13 illustrates the CFB decryption.

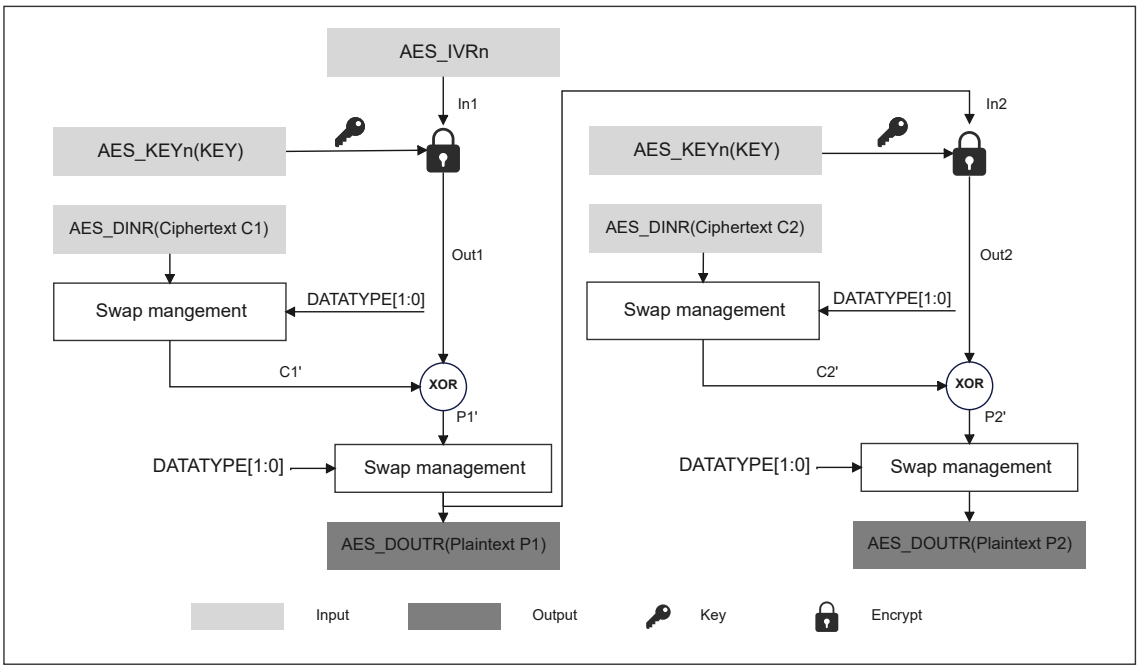


Figure 15-13 CFB Decryption

In CFB decrypt mode, like in ECB/CBC decrypt mode, the key must be prepared to perform an AES decryption.

After the key preparation process, the decryption goes as follows:

The first 128-bit ciphertext block (after the swap operation) is used, produce C1', the 128-bit IVI field as the AES core input block In1 for decrypt operation, using the 128-bit, 192-bit or 256-bit key. Its output Out1 is XOR-ed with C1' to produce the first plaintext block P1.

The second ciphertext block is processed in the same way as the first block, except that the P1 data from the first block is used in place of the initialization vector.

The decryption continues in this way until the last complete ciphertext block is decrypted.

15.3.6.1.4 OFB Encryption and Decryption

OFB Encryption

Figure 15-14 illustrates the Output Feedback (OFB) encryption.

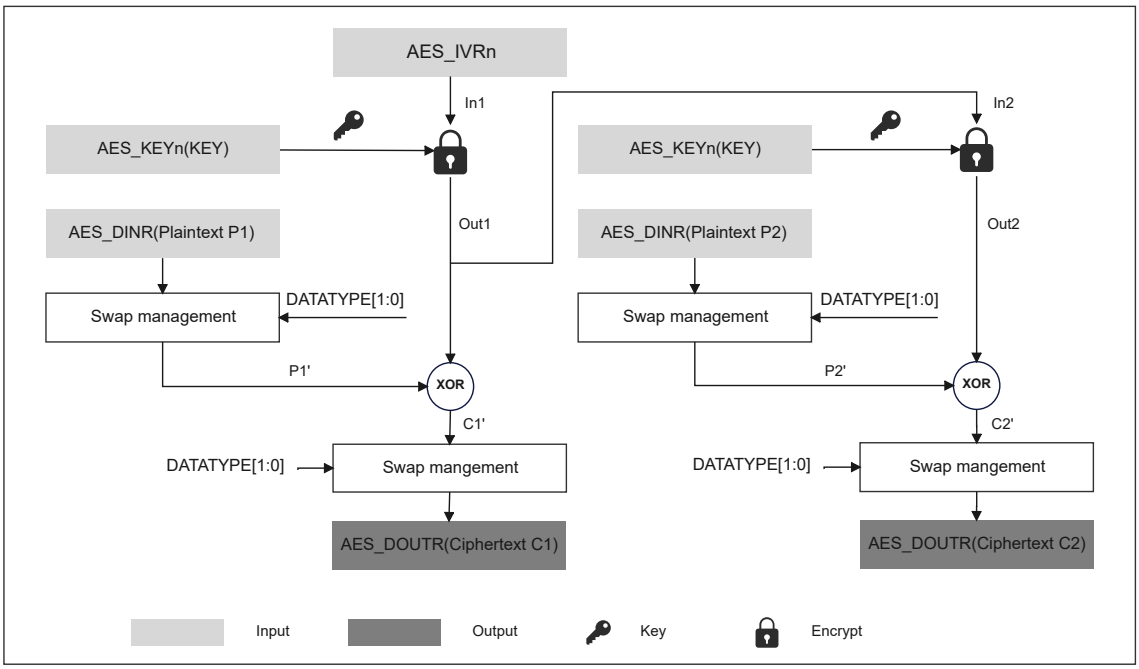


Figure 15-14 OFB Encryption

In OFB encrypt mode, the first plaintext input block produces the P1' after bit/byte/half-word swapping modules. a 128-bit IVI bitfield (initialization vector and counter) and a 128-bit,192-bit or 256-bit key produces Out1. P1' is XOR-ed with Out1, produce C1', which after bit/byte/half-word swapping as ciphertext C1.

The second ciphertext block is processed in the same way as the first block, except that the Out1 data from the first block is used in place of the initialization vector.

The chaining of data blocks continues in this way until the last plaintext block in the message is encrypted.

OFB Decryption

Figure 15-15 illustrates the OFB decryption.

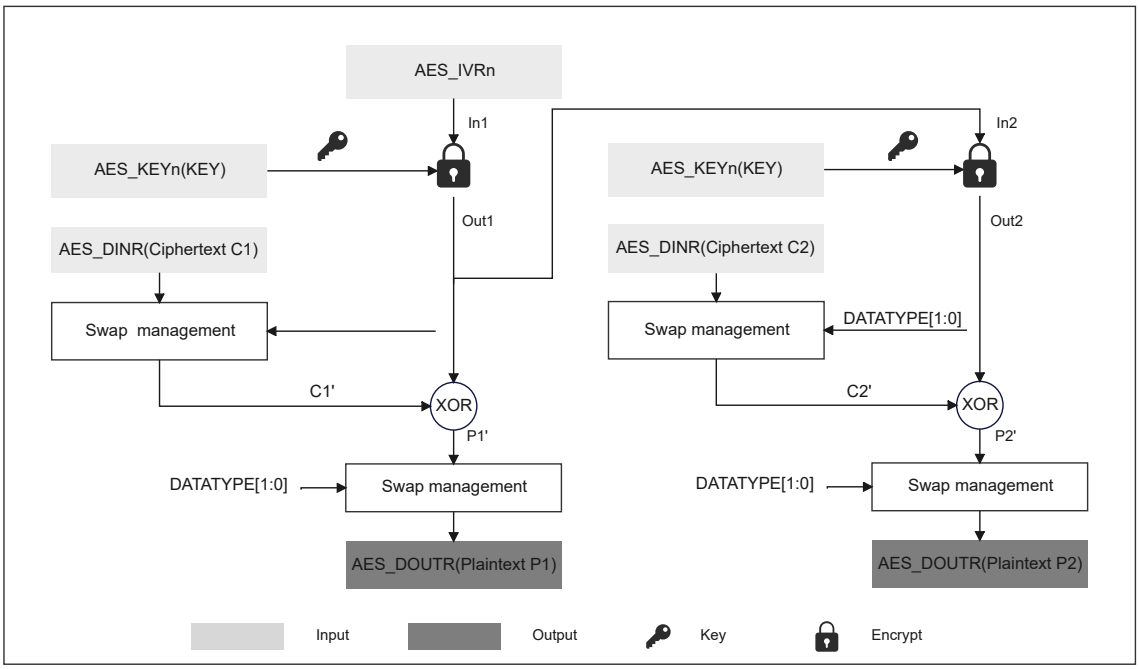


Figure 15-15 OFB Decryption

In OFB decrypt mode, like in ECB/CBC/CFB decrypt mode, the key must be prepared to perform an AES decryption.

After the key preparation process, the decryption goes as follows: the first 128-bit ciphertext block (after the swap operation) is used, produce C1', the 128-bit IVI field as the AES core input block In1 for decrypt operation, using the 128-bit, 192-bit or 256-bit key. Its output Out1 is XOR-ed with C1' to produce the first plaintext block P1.

The second ciphertext block is processed in the same way as the first block, except that the Out1 data from the first block is used in place of the initialization vector.

The decryption continues in this way until the last complete ciphertext block is decrypted.

15.3.6.2 Encryption and Decryption Sequence

Encryption Sequence

The sequence of events to perform an ECB/CBC encryption (more detailed information can be found in Encryption Process):

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select Mode 1 by setting the MODE bitfield of the AES_CR register to 00. Additionally, select ECB, CFB, OFB or CBC chaining mode by setting the CHMOD[2:0] bit field of the AES_CR register to 000 or 001, respectively. The data type can also be defined using the DATATYPE[1:0] bit field.
3. Select the key length of 128, 192, or 256 bits by using the KEYSIZE bit in the AES_CR register.
4. Write the AES_KEYRx registers (128, 192 or 256 bits) with encryption key. If CBC mode is selected, fill the AES_IVRx registers with the initialization vector data.
5. Enable the AES peripheral by setting the EN bit of the AES_CR register.
6. Write the AES_DINR register four times to input the plaintext (MSB first), as shown in Figure 15-16
7. Wait until the CCF flag is set in the AES_SR register.
8. Read the AES_DOUTR register four times to get the ciphertext (MSB first), as shown in Figure 15-16. Then, clear the CCF flag by setting the CCFC bit of the AES_CR register.

9. Repeat steps 6, 7, and 8 to process all the blocks with the same encryption key.

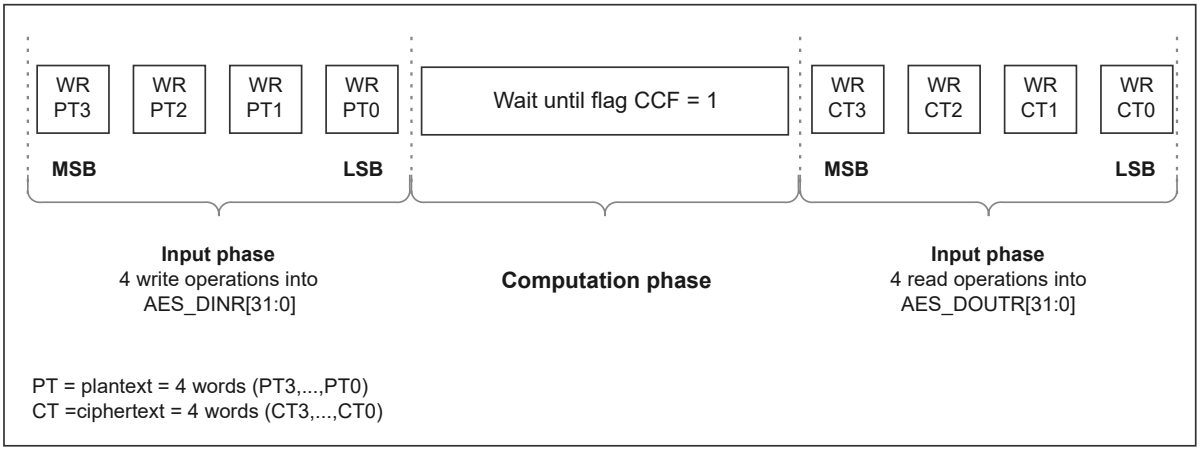


Figure 15-16 ECB/CBC/CFB/OFB Encryption (Mode 1)

Decryption Sequence

Here are the steps to perform AES ECB/CBC/CFB/OFB decryption:

1. Prepare the decryption key in the AES core by following the steps outlined in Decryption Round Key Preparation.
2. Select Mode 2 by setting the MODE bitfield of the AES_CR register to 1. Additionally, select ECB or CBC chaining mode by setting the CHMOD[2:0] bit field of the AES_CR register to 000 or 001, respectively. The data type can also be defined by using DATATYPE[1:0] bit field. KEYSIZE bitfield should remain unchanged.
3. Write the AES_IVRx registers with the initialization vector (required in CBC mode only).
4. Enable AES by setting the EN bit of the AES_CR register.
5. Write the AES_DINR register four times to input the cipher text (MSB first), as shown in Figure 15-17.
6. Wait until the CCF flag is set in the AES_SR register.
7. Read the AES_DOUTR register four times to get the plaintext (MSB first), as shown in Figure 15-17. Then, clear the CCF flag by setting the CCFC bit of the AES_CR register.
8. Repeat steps 5-6-7 to process all the blocks encrypted with the same key.

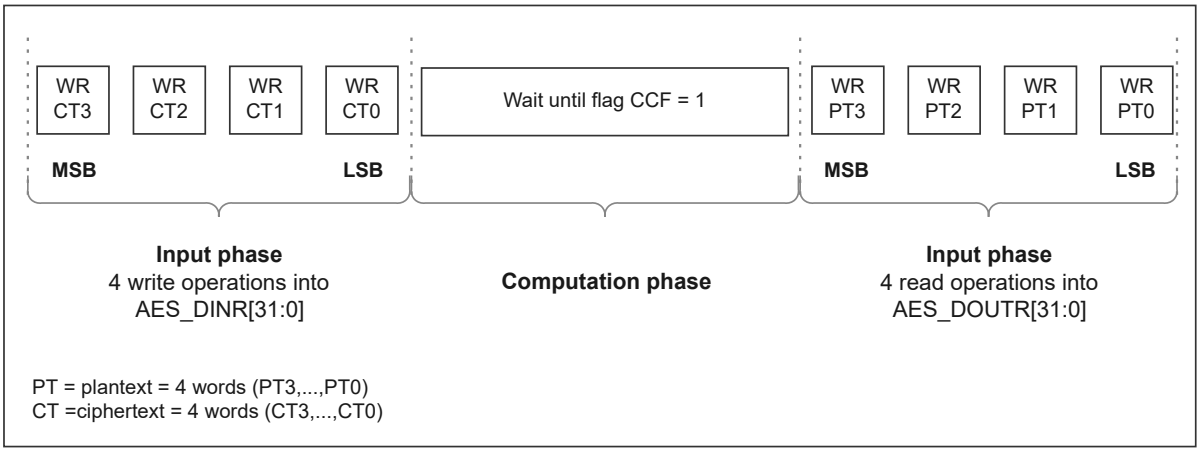


Figure 15-17 ECB/CBC Decryption (Mode 2)

15.3.6.3 Suspend/Resume Operations

Suspend Operations

To suspend the processing of a message, follow these steps:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES_CR register.
2. If DMA is not used, read four times the AES_DOUTR register to save the last processed block. However, if DMA is used, wait until the CCF flag is set in the AES_SR register, then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES_CR register.
3. If DMA is not used, the CCF flag of the AES_SR register should be polled until it becomes 1, indicating that the computation is completed.
4. Clear the CCF flag by setting the CCFC bit of the AES_CR register.
5. Set the ENMASK flag by setting the ENMASK bit of the AES_CR register.
6. Disable the AES peripheral by clearing the bit EN of the AES_CR register.
7. Save the AES_CR register and clear the key registers if not needed to process the higher priority message.

   NOTE: In this step, if the interrupted process is a decryption, the derived key information stored in AES_KEYRx registers can be optionally saved in memory. Otherwise, there is no need to save those registers as the original key value is known by the application.

8. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, and so on).

Resume Operations

To resume the processing of a message, follow these steps:

1. If DMA is used, configure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Restore the AES_CR register (with the correct KEYSIZE) then restore the AES_KEYRx registers. In the case of decryption, derived key information can be written in the AES_KEYRx register instead of the original key value.
4. Prepare the decryption key as described in (only required for ECB or CBC decryption).

   NOTE: This step is not necessary if derived key information is loaded in AES_KEYRx registers.

5. Restore AES_IVRx registers using the saved configuration (only required in CBC mode).
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.
7. If DMA is used, enable AES DMA transfers by setting the DMAINEN and DMAOUTEN bits of the AES_CR register.

15.3.7 Counter (CTR) Mode

15.3.7.1 Overview

The counter mode (CTR) uses AES as a key-stream generator. The generated keys are then XOR-ed with the plaintext to obtain the ciphertext.

The CTR chaining is defined in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation. A typical message construction in CTR mode is given in Figure 15-18.

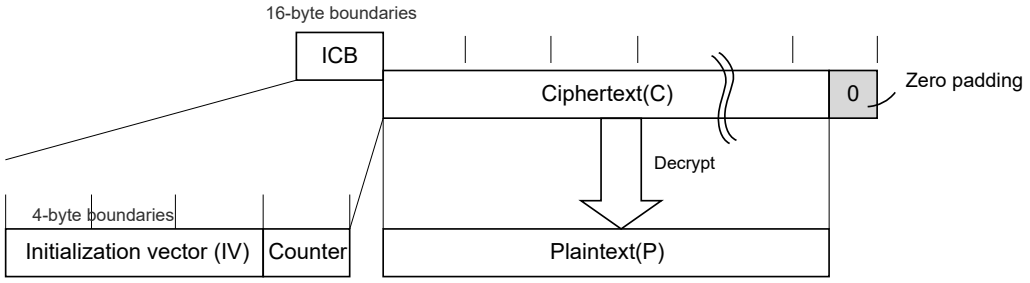


Figure 15-18 Message Construction in CTR Mode

The structure of this message is:

- A 16-byte Initial Counter Block (ICB), composed of two distinct fields:
  - Initialization Vector (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - Counter: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, and it has a known length. If the length is not a multiple of 16 bytes, then plaintext padding is necessary.

15.3.7.2 Encryption and Decryption

Figure 15-19 illustrates the CTR encryption process, as implemented in the AES peripheral.

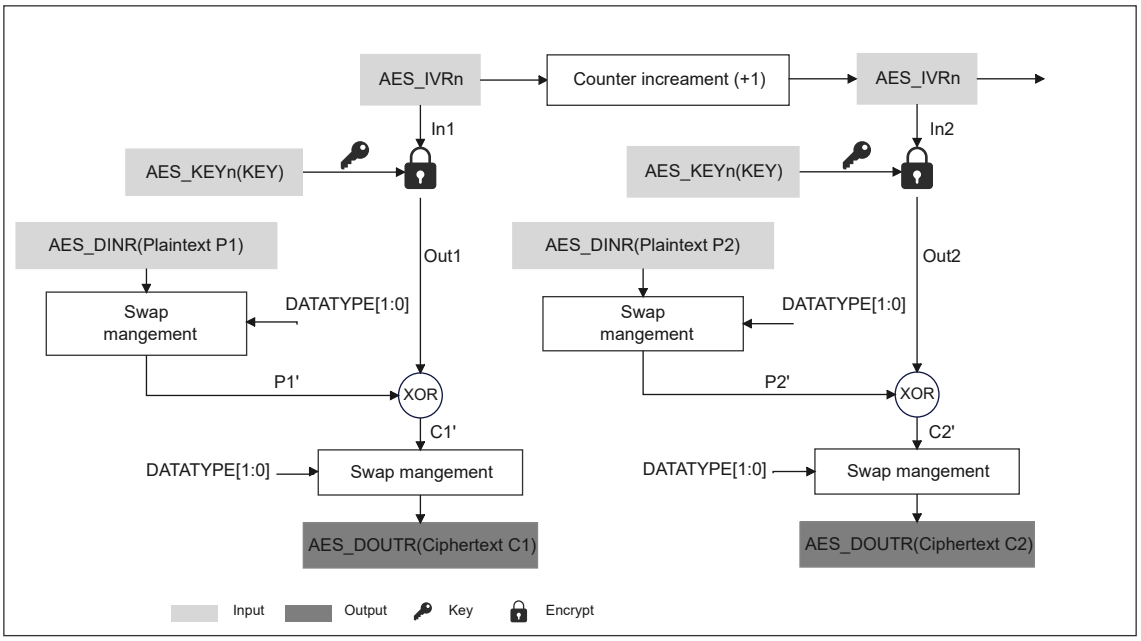


Figure 15-19 CTR Encryption

Figure 15-20 shows the CTR decryption process, as implemented in the AES peripheral.

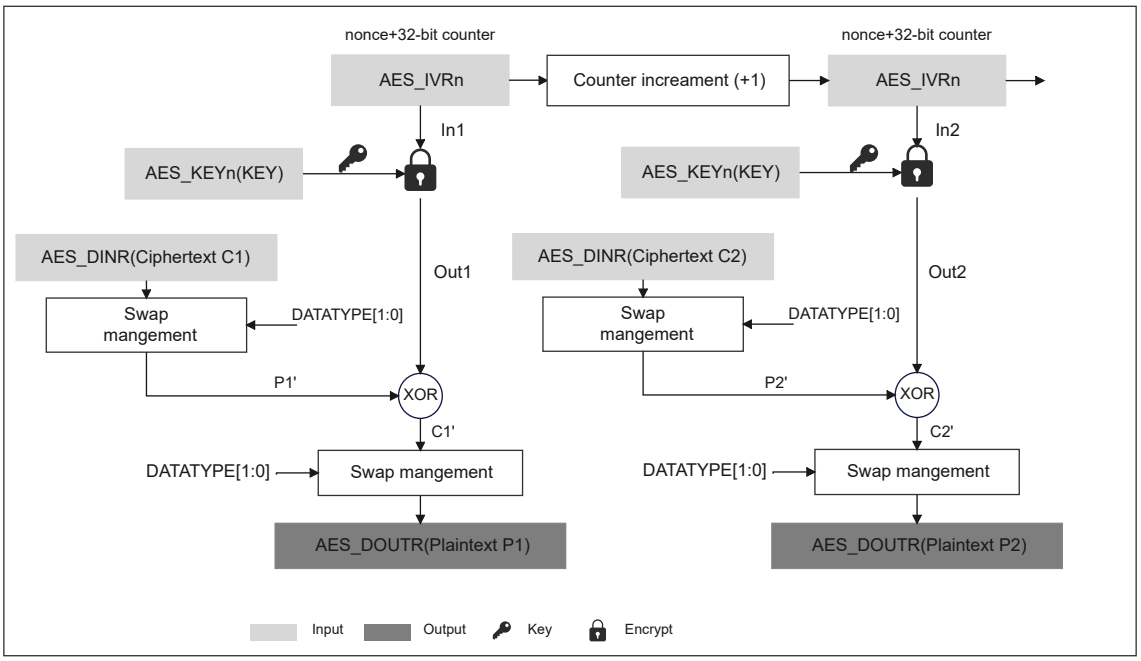


Figure 15-20 CTR Decryption

The CTR mode is selected by writing 010 to the CHMOD[2:0] bit field of the AES_CR register.

In CTR mode, the cryptographic core output (also called keystream) Outx is XOR-ed with relevant input block (Px' for encryption, Cx' for decryption), to produce the correct output block (Cx' for encryption, Px' for decryption). Initialization vectors in AES must be initialized as shown in Table 15-1.

Table 15-1 CTR Mode Initialization Vector Definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

Unlike in CBC mode which uses the AES_IVRx registers only once when processing the first data block, in CTR mode AES_IVRx registers are used for processing each data block, and the AES peripheral increments the counter bits of the initialization vector (leaving the nonce bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that is then XOR-ed with the plaintext (CTR encryption) or ciphertext (CTR decryption) input. In CTR mode, the MODE bitfield setting 1 (key derivation) is forbidden and all the other settings default to encryption mode.

The sequence of events to perform an encryption or decryption in CTR chaining mode:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select CTR chaining mode by setting to 010 the CHMOD[2:0] bit field of the AES_CR register.
3. Initialize the AES_KEYRx registers, and load the AES_IVRx registers as described in Table 15-1.
4. Set the EN bit of the AES_CR register, to start encrypting the current counter (EN is automatically reset when the calculation finishes).
5. If it is the last block, pad the data with zeros to have a complete block, if needed.
6. Append data in AES, and read the result. The three possible scenarios are described in the append data in AES and read the result. The three possible scenarios are described in Encryption Process.

7. Repeat the previous step till the second-last block is processed. For the last block, apply the two previous steps and discard the bits that are not part of the payload (if the size of the significant data in the last input block is less than 16 bytes).

15.3.7.3 Suspend/Resume Operations

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, and resume the message that was interrupted. Detailed CBC suspend/resume sequence is described in Chaining Modes.

NOTE: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.

15.3.8 Data Registers and Data Swapping

15.3.8.1 Data Input and Output

A 128-bit data block is entered into the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 128-bit data block is retrieved from the AES peripheral with four successive 32-bit word reads from the AES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The 32-bit data word for AES_DINR register or from AES_DOUTR register is organized in big endian order, that is:

- The most significant byte of a word to write into AES_DINR must be put on the lowest address out of the four adjacent memory locations keeping the word to write.
- The most significant byte of a word read from AES_DOUTR goes to the lowest address out of the four adjacent memory locations receiving the word.

For using DMA for input data block written into AES, the four words of the input block must be stored in the memory consecutively and in big-endian order, that is, the most significant word on the lowest address. See DMA Interface.

15.3.8.2 Data Swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0] bit field of the AES_CR register. The selection applies both to the input and the output of the AES core.

For different data swap types, Figure 15-21 shows the construction of AES processing core input buffer data P127 to P0, from the input data entered through the AES_DINR register, or the construction of the output data available through the AES_DOUTR register, from the AES processing core output buffer data P127 to P0.

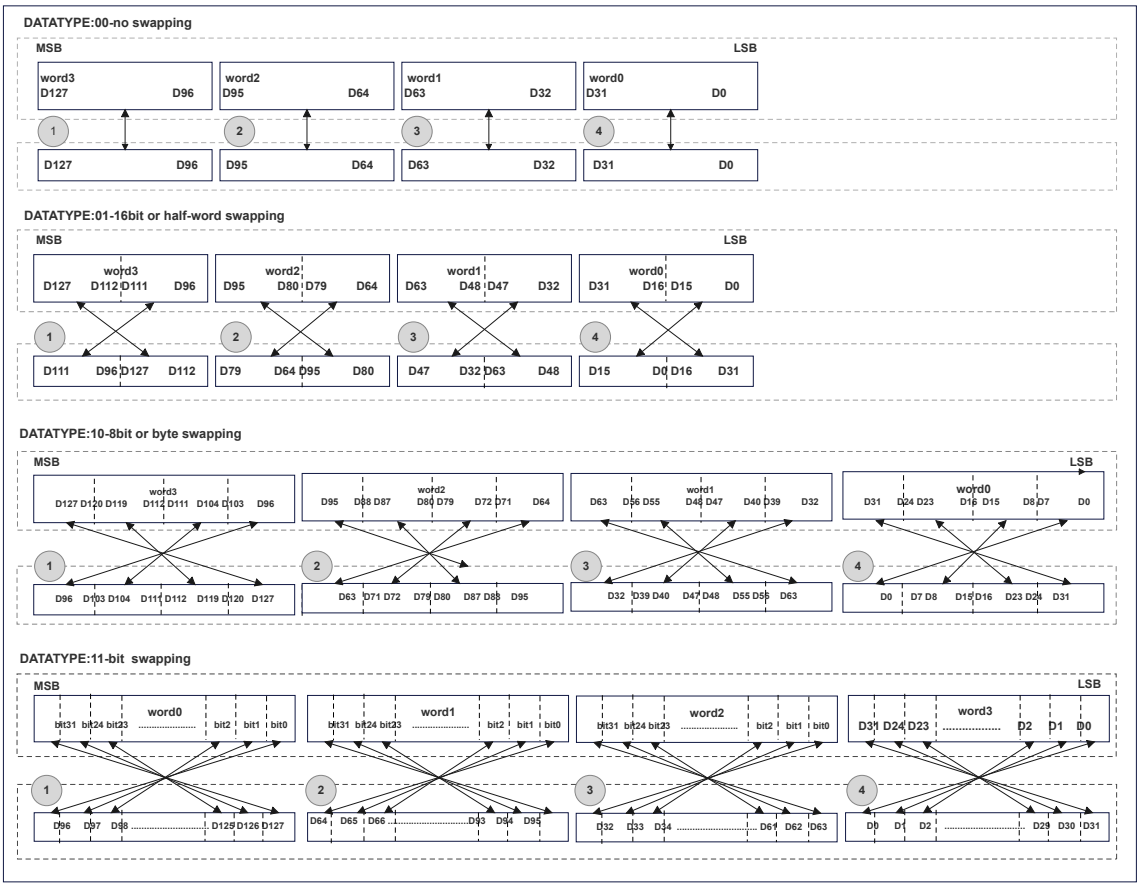


Figure 15-21 128-bit Block Construction of Data Swap

(1) MSB: Most significant bit (127) of memory data block /AES core buffer.

(2) LSB: Least significant bit (0) of memory data block /AES core buffer.

(3) Dx: Input/output data bit 'x'.

(4) ①~④: Order of write to AES_DINR/read from AES_DOUTR.

NOTE: The data in AES key registers (AES_KEYRx) and initialization registers (AES_IVRx) are not sensitive to the swap mode selection.

15.3.8.3 Data Padding

Figure 15-21 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 48 message bits, with DATATYPE[1:0] = 01
- 56 message bits, with DATATYPE[1:0] = 10
- 34 message bits, with DATATYPE[1:0] = 11

15.3.9 Key Registers

The AES_KEYRx registers store the encryption or decryption key bitfield KEY[127:0], KEY[192:0] or KEY[255:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address.

The key is spread over eight registers, as shown in Table 15-2.

Table 15-2 Key Endianness in AES_KEYRx Registers (128-Bit, 192-Bit or 256-Bit Key Length)

AES_KEYR7 [31:0]	AES_KEYR6[31:0]	AES_KEYR5[31:0]	AES_KEYR4[31:0]	AES_KEYR3[31:0]	AES_KEYR2 [31:0]	AES_KEYR1[31:0]	AES_KEYR0[31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
		KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key for encryption or decryption may be written into these registers when the AES peripheral is disabled by clearing the EN bit of the AES_CR register.

The key registers are not affected by the data swapping controlled by DATATYPE[1:0] bit field of the AES_CR register.

15.3.10 Initialization Vector Registers

The four AES_IVRx registers keep the initialization vector input bitfield IVI[127:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address. The registers are also ordered from lowest address (AES_IVR0) to highest address (AES_IVR3).

The signification of data in the bitfield depends on the chaining mode selected. When used, the bitfield is updated upon each computation cycle of the AES core.

When the AES peripheral is enabled, write operations to the AES_IVRx registers have no effect on the register contents. For modifying the contents of the AES_IVRx registers, the EN bit of the AES_CR register must first be cleared.

Reading the AES_IVRx registers returns the latest counter value (useful for managing suspend mode).

The AES_IVRx registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bit field of the AES_CR register.

15.3.11 DMA Interface

The AES peripheral provides an interface to connect to the Direct Memory Access (DMA) controller. The DMA operation is controlled through the AES_CR register.

15.3.11.1 Data Input Using DMA

Setting the DMAINEN bit of the AES_CR register enables DMA writing into AES. The AES peripheral then initiates a DMA request during the input phase each time it requires to write a 128-bit block (quadruple word) to the AES_DINR register, as shown in Figure 15-22.

NOTE: For further details on the possible requirement for special padding or ciphertext stealing, which depends on the selected algorithm and mode, refer to Encryption Process.

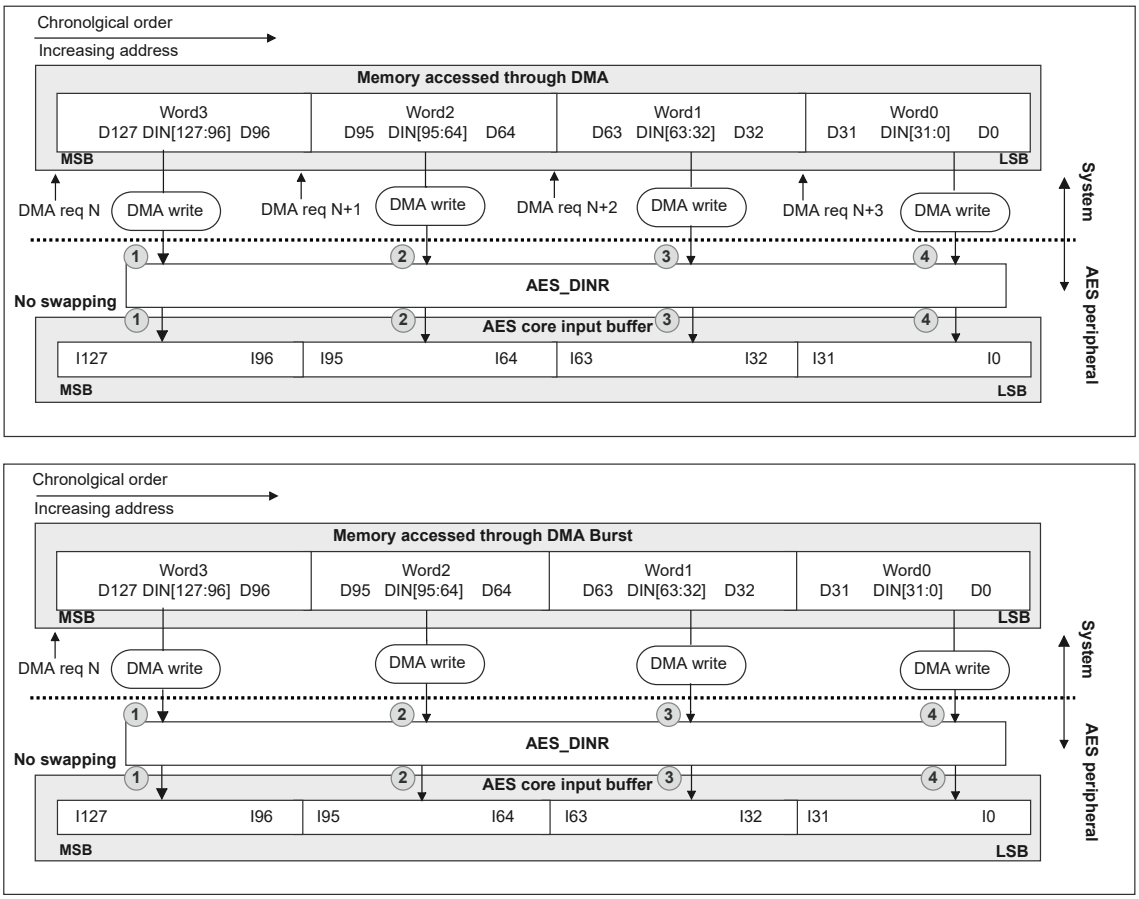


Figure 15-22 128-Bit Block Construction of Data Swap

(1) ① ~ ④: The order of write AES_DINR.

15.3.11.2 Data Output Using DMA

Setting the DMAOUTEN bit of the AES_CR register enables DMA reading from AES. The AES peripheral then initiates a DMA request during the Output phase each time it requires to read a 128-bit block (quadruple word) to the AES_DINR register, as shown in Figure 15-23.

NOTE: According to the message size, extra bytes might need to be discarded by application in the last block.

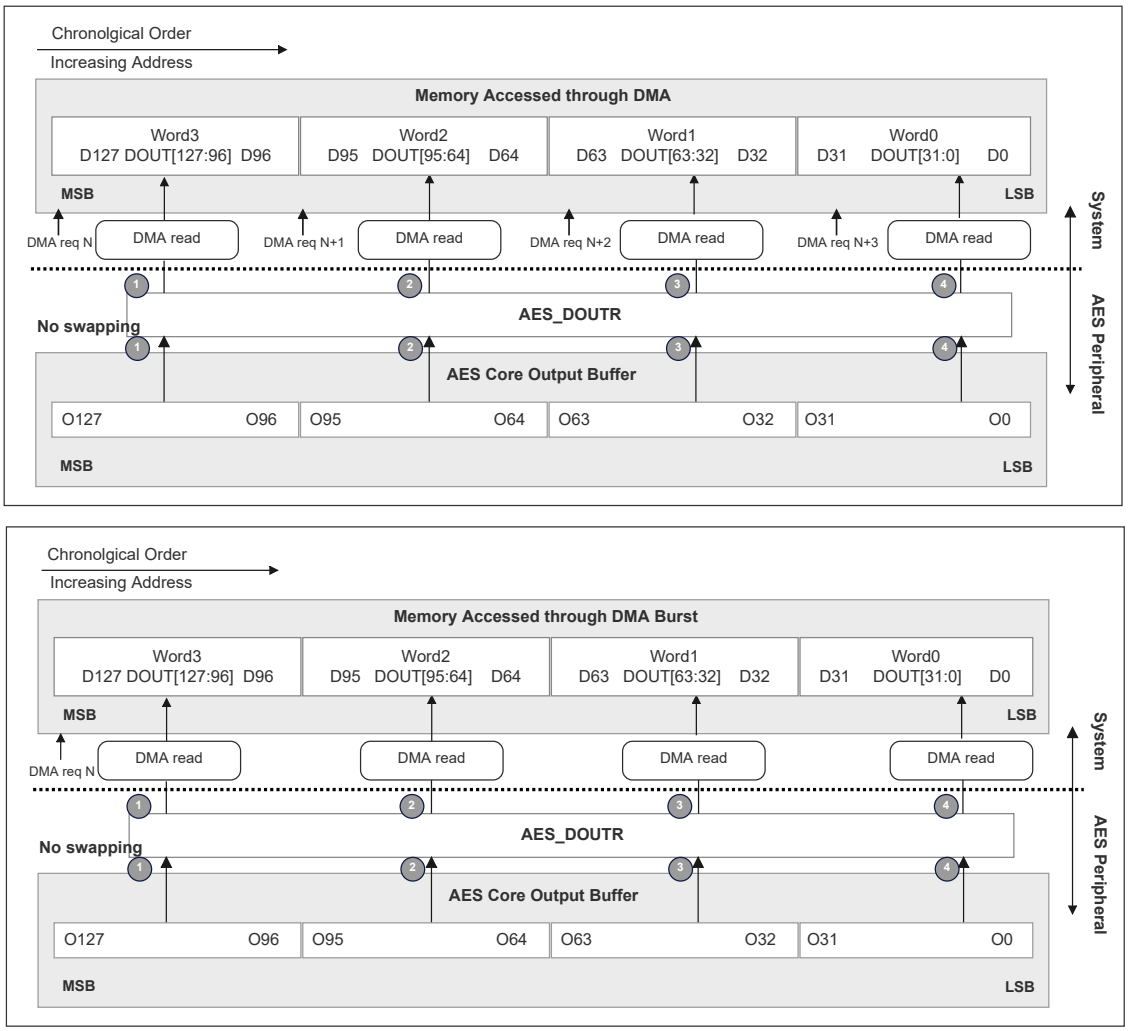


Figure 15-23 128-bit Block Construction of Data Swap

(1) ① ~ ④: The order of read AES_DOUTR.

15.3.11.3 DMA Operation in Different Operating Modes

DMA operations are usable when Mode 1 (encryption) or Mode 2 (decryption) are selected via the MODE bitfield of the register AES_CR. As in Mode 2 (key derivation) the AES_KEYRx registers must be written by software, enabling the DMA transfer through the DMAINEN and DMAOUTEN bits of the AES_CR register to have no effect in that mode.

DMA single requests are generated by AES until it is disabled. So, after the data output phase at the end of processing a 128-bit data block, AES switches automatically to a new data input phase for the next data block, if any.

When the data transferring between AES and memory is managed by DMA, the CCF flag is not relevant and can be ignored (left set) by software. It must only be cleared when transiting back to data transferring managed by software. See Suspend/Resume Operations as an example.

15.3.12 Error Management

15.3.12.1 Read Error Flag (RDERR)

Unexpected read attempt of the AES_DOUTR register sets the RDERR flag of the AES_SR register and returns zero.

RDERR is triggered during the computation phase or during the input phase.

NOTE: AES is not disabled upon an RDERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES_CR register is set. For more details, refer to Interrupts. The RDERR flag is cleared by setting the ERRC bit of the AES_CR register.

15.3.12.2 Write Error Flag (WDERR)

Unexpected write attempt of the AES_DINR register sets the WRERR flag of the AES_SR register, and has no effect on the AES_DINR register. The WRERR is triggered during the computation phase or the output phase.

NOTE: AES is not disabled after a WRERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES_CR register is set. For more details, refer to Interrupts. The WRERR flag is cleared by setting the ERRC bit of the AES_CR register.

15.4 Interrupts

The AES peripheral generates individual interrupt sources that signify different events, as mentioned in Table 15-3. These individual sources are subsequently combined into a common interrupt signal that connects to the NVIC. Each source can be individually enabled or disabled by manipulating the corresponding enable bit in the AES_CR register. The status of each source can be obtained by reading the AES_SR register. Table 15-3 provides a summary of the interrupt sources, their event flags, and enable bits.

Table 15-3 AES Interrupt Requests

Interrupt Acronym	AES Interrupt Event	Event Flag	Enable Bit	Interrupt Clear Method
AES	Computation completed flag	CCF	CCFIE	Set CCFC
	Read error flag	RDERR	ERRIE	Set ERRC
	Write error flag	WRERR		

15.5 Processing Latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

Table 15-4 Processing Latency for ECB, CBC and CTR

Key Size	Mode of Operation	Algorithm	Clock Cycles
128-bit	Mode 1: Encryption	ECB, CBC, CTR, CFB, OFB	19
	Mode 2: Decryption	ECB, CBC, CTR, CFB, OFB	19
192-bit	Mode 1: Encryption	ECB, CBC, CTR, CFB, OFB	21
	Mode 2: Decryption	ECB, CBC, CTR, CFB, OFB	21
256-bit	Mode 1: Encryption	ECB, CBC, CTR, CFB, OFB	23
	Mode 2: Decryption	ECB, CBC, CTR, CFB, OFB	23

Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

15.6 Registers

15.6.1 Register Address Map

Offset	Register Name	Register Description
0x0000	AES_CR	AES control register
0x0004	AES_SR	AES status register
0x0008	AES_DINR	AES data input register
0x000c	AES_DOUT	AES data output register
0x0010	AES_KEY0	AES key register 0
0x0014	AES_KEY1	AES key register 1
0x0018	AES_KEY2	AES key register 2
0x001c	AES_KEY3	AES key register 3
0x0020	AES_IVR0	AES initialization vector register 0
0x0024	AES_IVR1	AES initialization vector register 1
0x0028	AES_IVR2	AES initialization vector register 2
0x002c	AES_IVR3	AES initialization vector register 3
0x0030	AES_KEY4	AES key register 4
0x0034	AES_KEY5	AES key register 5
0x0038	AES_KEY6	AES key register 6
0x003c	AES_KEY7	AES key register 7

15.6.2 Register Field Details

15.6.2.1 AES_CR

0x0000			AES control register											AES_CR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved							ENMASK	NPBLB				KEYSIZE		Reserved	
Type	RO							RW	RW				RW		RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CCFC	ODATSWAP	IDATSWAP	DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CHMOD			Reserved	MODE	DATATYPE		EN
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW			RO	RW	RW		RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-5 AES Control Register Description

Field	Name	Description
31:25	Reserved	Reserved
24	ENMASK	AES enable mask 0: The key will be discarded when the EN bit of the AES_CR register is clear, users need to configure the key again. 1: The key will be kept when the EN bit of the AES_CR register is clear. The AES will process the next block using the previous intermediate result if ENMASK = 1 and EN = 0 followed by EN = 1.
23:20	NPBLB	Number of padding bytes in last block The bitfield sets the number of padding bytes in the last block of payload: 0000: All bytes are valid (no padding)

Field	Name	Description
		0001: Padding for one least-significant byte of last block ... 1111: Padding for 15 least-significant bytes of last block
19:18	KEYSIZE	Key size selection This bitfield defines the length of the key used in the AES cryptographic core, in bits: 00: 128 01: 192 10: 256 Attempts to write the bit are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.
17:16	Reserved	Reserved
15	CCFC	Computation complete flag clear Upon written to 1, this bit clears the computation complete flag (CCF) in the AES_SR register: 0: No effect 1: Clear CCF Reading this bit always returns zero.
14	ODATSWAP	Output data swap 0: Output data no change 1: Output data swap based on DATATYPE
13	IDATSWAP	Input data swap 0: Input data no change 1: Input data swap based on DATATYPE
12	DMAOUTEN	DMA output enable

Field	Name	Description
		This bit enables/disables data transferring with DMA in the output phase: 0: Disable 1: Enable When this bit is set, DMA requests are automatically generated by AES during the output data phase.
11	DMAINEN	DMA input enable This bit enables/disables data transferring with DMA in the input phase: 0: Disable 1: Enable When this bit is set, DMA requests are automatically generated by AES during the input data phase.
10	ERRIE	Error interrupt enable This bit enables or disables (masks) the AES interrupt generation when RDERR and/or WRERR is set: 0: Disable (mask) 1: Enable
9	CCFIE	CCF interrupt enable This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set: 0: Disable (mask) 1: Enable
8	ERRC	Error flag clear Upon written to 1, this bit clears the RDERR and WRERR error flags in the AES_SR register: 0: No effect 1: Clear RDERR and WRERR flags Reading the flag always returns zero.

Field	Name	Description
7:5	CHMOD	Chaining mode selection This bitfield selects the AES chaining mode: 000: Electronic codebook (ECB) 001: Cipher-block chaining (CBC) 010: Counter mode (CTR) 011: Cipher feedback mode (CFB) 100: Output feedback mode (OFB) Others: Reserved Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access, and it is not cleared by that write access.
4	Reserved	Reserved
3	MODE	AES operating mode This bitfield selects the AES operating mode: 0: Encryption 1: Decryption Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.
2:1	DATATYPE	Data type selection This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register by selecting the mode of data swapping: 00: None 01: Half-word (16-bit) 10: Byte (8-bit) 11: Bit

Field	Name	Description
		For more details, refer to Data Registers and Data Swapping. Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access, and it is not cleared by that write access.
0	EN	AES enable This bit enables/disables the AES peripheral: 0: Disable 1: Enable At any moment, clearing and then setting the bit re-initializes the AES peripheral. This bit is automatically cleared by hardware upon completing the key preparation (Mode 2).

15.6.2.2 AES_SR

0x0004	AES status register													AES_SR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												BUSY	WRERR	RDERR	CCF
Type	RO												RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-6 AES Status Register Description

Field	Name	Description
31:4	Reserved	Reserved
3	BUSY	Busy This flag indicates whether AES is idle or busy during the GCM payload encryption phase: 0: Idle 1: Busy When the flag indicates 'idle', the current GCM encryption processing may be suspended to process a higher-priority message. In other chaining modes, or in GCM phases other than payload encryption, the flag must be ignored for the suspend process.
2	WRERR	Write error

Field	Name	Description
		This flag indicates the detection of an unexpected write operation to the AES_DINR register (during computation or data output phase): 0: Not detected 1: Detected The flag is set by hardware. It is cleared by software upon setting the ERRC bit. Upon the flag set, an interrupt is generated if ERRI E= 1. The flag setting has no impact on the AES operation. Unexpected writing is ignored.
1	RDERR	Read error flag This flag indicates the detection of an unexpected read operation from the AES_DOUTR register (during computation or data input phase): 0: Not detected 1: Detected The flag is set by hardware. It is cleared by software upon setting the ERRC bit. Upon the flag set, an interrupt is generated if ERRIE = 1. The flag setting has no impact on the AES operation. Unexpected read returns zero.
0	CCF	Computation completed flag This flag indicates whether the computation is completed: 0: Not completed 1: Completed The flag is set by hardware upon the completion of the computation. It is cleared by software, upon setting the CCFC bit. Upon the flag set, an interrupt is generated if CCFIE = 1. The flag is significant only when the DMAOUTEN bit is 0. It may stay high when DMAEN is 1.

15.6.2.3 AES_DINR

0x0008			AES data input register											AES_DINR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DIN															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DIN															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-7 AES Data Input Register Description

Field	Name	Description
31:0	DIN	Input data word A four-fold sequential write to this bitfield during the input phase results in writing a complete 128-bit block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 128-bit input buffer. The data signification of the input data block depends on the AES operating mode: • Mode 0 (encryption): Plaintext • Mode 1 (decryption): Ciphertext

15.6.2.4 AES_DOUT

0x000c			AES data output register											AES_DOUT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DOUT															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DOUT															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-8 AES Data Output Register Description

Field	Name	Description
31:0	DOUT	Output data word This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF set), virtually reads a complete 128-bit block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield. Data weights from the first to the fourth read operation are [127:96], [95:64], [63:32], and [31:0]. The data signification of the output data block depends on the AES operating mode: • Mode 0 (encryption): Ciphertext • Mode 1 (decryption): Plaintext

15.6.2.5 AES_KEY0

0x0010			AES key register 0											AES_KEY0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-9 AES Key Register 0 Description

Field	Name	Description
31:0	KEY0	KEY[31:0]: Cryptographic key, bits [31:0]

15.6.2.6 AES_KEY1

0x0014			AES key register 1											AES_KEY1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-10 AES Key Register 1 Description

Field	Name	Description
31:0	KEY1	KEY[63:32]: Cryptographic key, bits [63:32]

15.6.2.7 AES_KEY2

0x0018			AES key register 2											AES_KEY2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY2															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY2															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-11 AES Key Register 2 Description

Field	Name	Description
31:0	KEY2	KEY[95:64]: Cryptographic key, bits [95:64]

15.6.2.8 AES_KEY3

0x001c			AES key register 3											AES_KEY3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY3															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY3															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-12 AES Key Register 3 Description

Field	Name	Description
31:0	KEY3	KEY[127:96]: Cryptographic key, bits [127:96]

15.6.2.9 AES_IVR0

0x0020			AES initialization vector register 0											AES_IVR0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	IV0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IV0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-13 AES Initialization Vector Register 0 Description

Field	Name	Description
31:0	IV0	IVI[31:0]: Initialization vector input, bits [31:0]

15.6.2.10 AES_IVR1

0x0024			AES initialization vector register 1											AES_IVR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	IV1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IV1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-14 AES Initialization Vector Register 1 Description

Field	Name	Description
31:0	IV1	IVI[63:32]: Initialization vector input, bits [63:32]

15.6.2.11 AES_IVR2

0x0028			AES initialization vector register 2											AES_IVR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	IV2															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IV2															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-15 AES Initialization Vector Register 2 Description

Field	Name	Description
31:0	IV2	IVI[95:64]: Initialization vector input, bits [95:64]

15.6.2.12 AES_IVR3

0x002c			AES initialization vector register 3											AES_IVR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	IV3															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	IV3															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-16 AES Initialization Vector Register 3 Description

Field	Name	Description
31:0	IV3	IVI[127:96]: Initialization vector input, bits [127:96]

15.6.2.13 AES_KEY4

0x0030			AES key register 4											AES_KEY4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY4															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY4															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-17 AES Key Register 4 Description

Field	Name	Description
31:0	KEY4	KEY[31:0]: Cryptographic key, bits [31:0]

15.6.2.14 AES_KEY5

0x0034			AES key register 5											AES_KEY5		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY5															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY5															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-18 AES Key Register 5 Description

Field	Name	Description
31:0	KEY5	KEY[63:32]: Cryptographic key, bits [63:32]

15.6.2.15 AES_KEY6

0x0038			AES key register 6											AES_KEY6		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY6															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY6															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-19 AES Key Register 6 Description

Field	Name	Description
31:0	KEY6	KEY[95:64]: Cryptographic key, bits [95:64]

15.6.2.16 AES_KEY7

0x003c			AES key register 7											AES_KEY7		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	KEY7															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KEY7															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-20 AES Key Register 7 Description

Field	Name	Description
31:0	KEY7	KEY[127:96]: Cryptographic key, bits [127:96]

Hash Processor (HASH)

This chapter describes the details of the Hash Processor (HASH).

Topics: **Page**

16.1 Introduction

The Hash Processor (HASH) is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-224, SHA-256), the MD5 (message-digest algorithm 5) hash algorithm algorithm suitable for a variety of applications.

The HASH computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of up to (2^{64} – 1) bits. It also computes 128 bits digests for the MD5 algorithm.

16.2 Features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, (SHA-1 and SHA-2 family)
  - Internet Engineering Task Force (IETF) Request For Comments RFC 1321
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Word swapping supported: bits, bytes, half-words and 32-bit words
- Automatic padding to complete the input bit string to fit digest minimum block size of 512 bits (16 × 32 bits)
- Single 32-bit input register associated to an internal input FIFO of sixteen 32-bit words, corresponding to one block size
- Fast computation of SHA-1, SHA-224, SHA-256, and MD5
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 68 clock cycles for processing one 512-bit block of data using MD5 algorithm
- AHB slave peripheral, accessible through 32-bit word accesses only (else an AHB error is generated)
- 8 × 32-bit words (H0 to H7) for output message digest
- Automatic data flow control with support of Direct Memory Access (DMA) using one channel. Fixed burst of 4 supported.

16.3 Functional Description

16.3.1 Block Diagram

Figure 16-1 shows the block diagram of the Hash Processor (HASH).

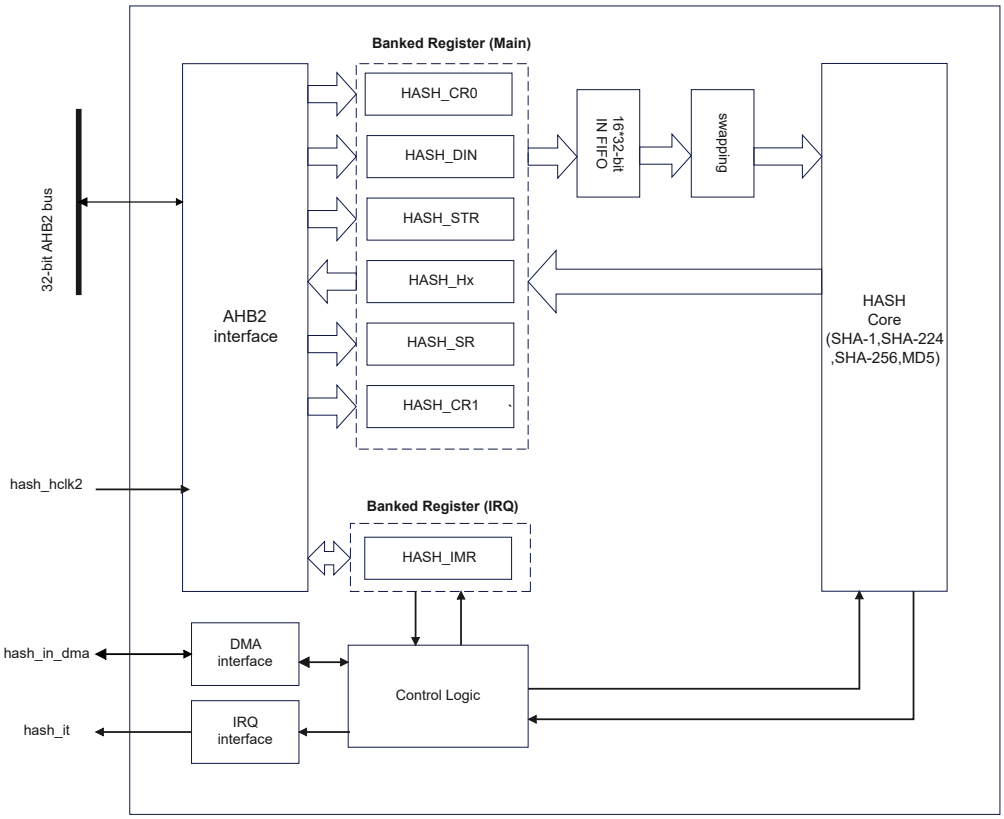


Figure 16-1 HASH Block Diagram

16.3.2 Internal Signal

Table 16-1 describes a list of the internal signals available at the HASH level, not at the product level (on pads).

Table 16-1 HASH Internal Input/Output Signals

Signal Name	Signal Type	Description
hash_hclk2	digital input	AHB2 bus clock
hash_it	digital output	Hash processor global interrupt request
hash_in_dma	digital input/output	DMA burst request/acknowledge

16.3.3 Secure Hash Algorithm

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard and the IETF RFC1321 publication (MD5).

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message of any length below 2^{64} bits is provided on input, the SHA-1, SHA-224, SHA-256 and MD5 processing core produces respectively a 160-bit, 224 bit, 256 bit and 128-bit output string called a message digest. The message digest can then be processed with a digital signature algorithm to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The verifier of a

digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” because it is computationally infeasible to find a message that corresponds to a given message digest (SHA-1 is no more qualified as secure since February 2017), or to find two different messages that produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.

16.3.4 Message Data Feeding

The message (or data file) to be processed by the HASH should be considered as a bit string. Per FIPS PUB 180-1 and 180-2 standards, this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example, message string “abc” with a bit string representation of “01100001 01100010 01100011” is represented by a 32-bit word 0x00636261 and 8-bit words 0x61626300.

Data are entered into the HASH one 32-bit word at a time by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO (IN FIFO) each time the register is written with new data. Hence, HASH_DIN and the input FIFO form a seventeen 32-bit word length FIFO (named the IN buffer).

In accordance with the kind of data to be processed (for example, byte swapping when data are ASCII text stream), there must be a bit, byte, half-word, or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. Figure 16-2 shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit word popped into IN FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR0).

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = ”00”) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

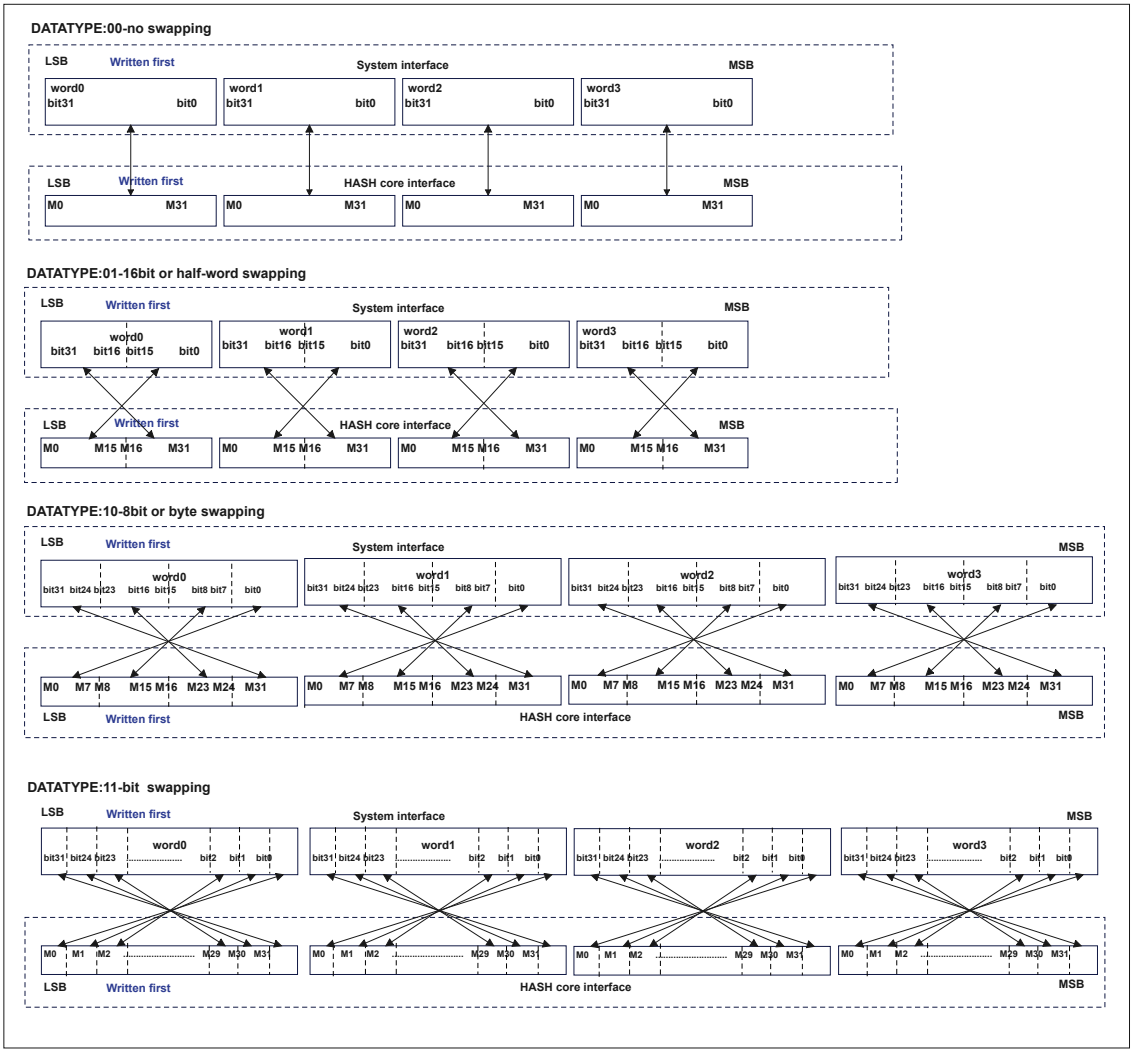


Figure 16-2 Message Data Swapping Feature

16.3.5 Message Digest Computing

The hash processor sequentially processes 512-bit blocks when computing the message digest. Thus, each time 16 × 32-bit words (= 512 bits) have been written to the hash processor by the DMA or the CPU, the HASH automatically starts computing the message digest. This operation is known as ‘partial digest computation’.

As described in Message Data Feeding, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO. To perform the hash computation on this data, the below sequence shall be used by the application.

1. Initialize the hash processor using the HASH_CR0 register:
   - Select the right algorithm using the ALGSEL field. If needed, program the correct swapping operation on the message input words using the DATATYPESEL bitfield in HASH_CR0.
   - Update BITNUMLW to define the number of valid bits in the last word if it differs from 32 bits. If this is the case, automatic padding could be applied by the HASH.
2. Complete the initialization by setting to 1 the INIT bit in HASH_CR0. Also, set the bit DMAEN to 1 if data are transferred via DMA.

NOTE: When programming step 2, it is important to set up before or at the same time the correct configuration values (ALGSEL, DATATYPESEL).

3. Start filling data by writing to the HASH_DIN register unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the IN FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   - When data are filled by software:
     - The partial digest computation is triggered when the software writes an additional word to the HASH_DIN register (actually, the first word of the next block). Once the processor is ready again (DIGCALSTAT = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.
     - The final digest computation is triggered when the last block is entered, and the software writes the DIGCAL bit to 1. If the message length is not an exact multiple of 512 bits, the BITNUMLW field in the HASH_STR register must be written prior to writing the DIGCAL bit (refer to Message Padding for details).
   - When data are filled by DMA:
     - The partial digest computation is triggered when an additional word is written to the HASH_DIN register.
     - The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register (the DIGCAL bit is set to 1 by hardware). If the message length is not an exact multiple of 512 bits, the BITNUMLW field in the HASH_STR register must be written prior to enabling the DMA (refer to Message Padding for details).
4. Once computed, the digest can be read from the output registers as described in Table 16-2.

Table 16-2 Hash Processor Outputs

Algorithm	Valid Output Registers	Most Significant Bit	Digest Size (in Bits)
MD5	HASH_H0 to HASH_H3	HASH_H0[31]	128
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA-256	HASH_H0 to HASH_H7	HASH_H0[31]	256

16.3.6 Message Padding

16.3.6.1 Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every 512-bit block) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, BITNUMLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

16.3.6.2 Padding Processing

Detailed padding sequences with DMA is enabled or disabled are described in Message Digest Computing.

16.3.6.3 Padding Example

As specified by Federal Information Processing Standards PUB 180-1 and PUB 180-2, message padding consists of appending a “1” followed by k “0”s, itself followed by a 64-bit integer that is equal to the length in bits of the message. These three padding operations generate a padded message of length L + 1 + k + 64, which by construction is a multiple of 512 bits.

Example from FIPS PUB180-2

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length

L = 24:

```
byte 0   byte 1  byte 2  byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

BITNUMLW has to be loaded with the value 24: a “1” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The message length value, L, in two-word format (that is `00000000 00000018`) is appended. Hence the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR0), the above message has to be entered by following below the sequence:

1. `0xUU636261` is written to the HASH_DIN register (where ‘U’ means don’t care).
2. `0x18` is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits).
3. `0x8000` is written to the HASH_STR register to start the message padding (described above) and then perform the digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0...4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

   ```
   HASH_H0 = 0xA9993E36
   HASH_H1 = 0x4706816A
   HASH_H2 = 0xBA3E2571
   HASH_H3 = 0x7850C26C
   ```

```
HASH_H4 = 0x9CD0D89D
```

16.3.7 HASH DMA Interface

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAEN bit in the HASH_CR0 register. When this bit is set, the HASH asserts the burst request signal to the DMA controller when there are enough free words in the FIFO to support a burst of four words.

Once four 32-bit words have been received, the HASH automatically restarts this process, checks the FIFO size, and asserts a new request if the FIFO status allows a burst reception. For more information, refer to Message Digest Computing.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that will be copied into the HASH_DIN register. This is done by writing the following value into the HASH_STR register: BITNUMLW = Len(Message)% 32, where “x%32” gives the remainder of x divided by 32.

16.3.8 HASH Error Management

No error flags are generated by the HASH hardware.

16.4 Interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal following events:

- Digest calculation completion (DIGCALSTAT)
- Data input buffer ready (DATAINSTAT)

Both interrupt sources are connected to the same global interrupt request signal, as shown on Figure 16-3.

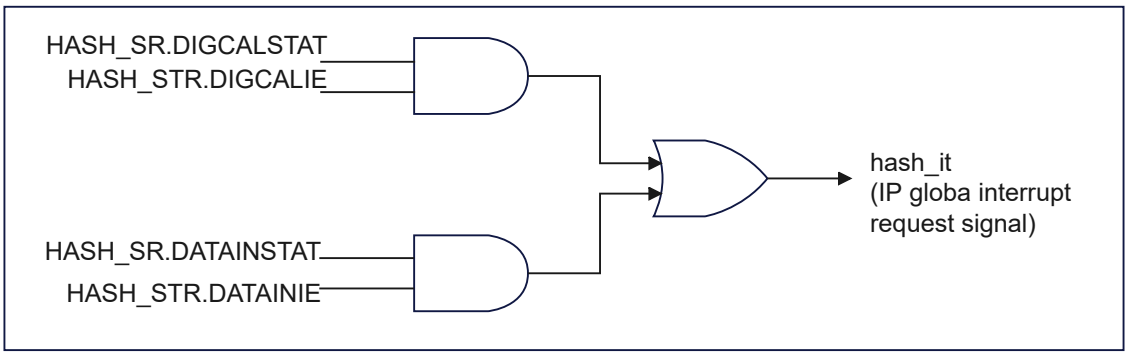


Figure 16-3 HASH Interrupt Mapping Diagram

The above interrupt sources can be enabled or disabled individually by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of the individual interrupt events can be read from the HASH_SR register. Table 16-3 summarizes the available features.

Table 16-3 HASH Interrupt Requests

Interrupt Event	Event Flag	Enable Control Bit
Digest computation completed flag	DIGCALSTAT	DIGCALIE

Interrupt Event	Event Flag	Enable Control Bit
Data input buffer ready to get a new block flag	DATAINSTAT	DATAINIE

16.5 Processing Time

Table 16-4 summarizes the time required to process a 512-bit intermediate block for each mode of operation.

Table 16-4 Processing Time (in Clock Cycle)

Mode of Operation	FIFO Load (1)	Computation Phase	Total
MD5	16	52	68
SHA-1	16	66	82
SHA-224	16	52	68
SHA-256			

(1) The time required to load the 16 words of the block into the processor must be added to this value.

The time required to process the last block of a message can be longer. This time depends on the length of the last block.

Compared to the processing of an intermediate block, it can be increased by the factor below:

- 1 to 2.5 for a hash message

16.6 Registers

16.6.1 Register Address Map

Offset	Register Name	Register Description
0x0000	HASH_CR0	Hash control register 0
0x0004	HASH_DIN	Hash data register
0x0008	HASH_STR	Hash start register
0x000c	HASH_H0	Hash digest register0
0x0010	HASH_H1	Hash digest register1
0x0014	HASH_H2	Hash digest register2
0x0018	HASH_H3	Hash digest register3
0x001c	HASH_H4	Hash digest register4
0x0020	HASH_H5	Hash digest register5
0x0024	HASH_H6	Hash digest register6
0x0028	HASH_H7	Hash digest register7
0x002c	HASH_IMR	Hash interrupt enable register
0x0030	HASH_SR	Hash status register
0x0034	HASH_CR1	Hash control register 1

16.6.2 Register Field Details

16.6.2.1 HASH_CR0

0x0000			Hash control register 0											HASH_CR0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved									ALGSEL		BURST	DATATYPESEL		DMAEN	INIT
Type	RO									RW		RW	RW		RW	WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-5 Hash Control Register 0 Description

Field	Name	Description
31:7	Reserved	Reserved
6:5	ALGSEL	Algorithm selection 0x0: Select SHA1 0x1: Select MD5 0x2: Select SHA224 0x3: Select SHA256
4	BURST	Burst operation 0x0: One DMA request for one data 0x1: One DMA request for four data
3:2	DATATYPESEL	Data type selection 0x0: 32-bit data. 32-bit data written to DIN do not need swap locations.

Field	Name	Description
		0x1: 16-bit data. 32-bit data written to DIN is treated as two and a half words, and they will be swapped. 0x2: 8-bit data. 32-bit data written to DIN is treated as four bytes, and they will be swapped. 0x3: 1-bit data. Data written to DIN is treated as 32-bit, and they will be swapped.
1	DMAEN	DMA enable 0x0: Disable DMA transfer 0x1: Enable DMA transfer
0	INIT	Initialize message digest calculation Writing 1 will reset the hash core so that the message digest of the new message can be calculated. Writing 0 will not affect it, and reading the bit will always return 0.

16.6.2.2 HASH_DIN

0x0004			Hash data register											HASH_DIN		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DIN															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DIN															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-6 Hash Data Register Description

Field	Name	Description
31:0	DIN	Hash data input register

16.6.2.3 HASH_STR

0x0008			Hash start register											HASH_STR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DIGCAL	Reserved										BITNUMLW				
Type	WC	RO										RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-7 Hash Start Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	DIGCAL	Digest calculation Writing 1 will start populating message with the BITNUMLW value previously written and then will start the final message digest calculation. Reading the bit will always return 0.
14:5	Reserved	Reserved
4:0	BITNUMLW	Number of valid bits in the last word. 0x00: All 32-bit data in the last word are valid. That is DIN[31:0] 0x01: Only one bit of the last word is valid. That is DIN[31]. 0x02: Only two bits of the last word are valid. That is DIN[31:30]. 0x03: Only three bits of the last word are valid. That is DIN[31:29]. ...

Field	Name	Description
		0x1F: Only 32 bits of the last word are valid. That is DIN[31:1].

16.6.2.4 HASH_H0

0x000c			Hash digest register 0											HASH_H0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DR0															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR0															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-8 Hash Digest Register 0 Description

Field	Name	Description
31:0	DR0	Hash digest register 0

16.6.2.5 HASH_H1

0x0010			Hash digest register 1											HASH_H1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DR1															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR1															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-9 Hash Digest Register 1 Description

Field	Name	Description
31:0	DR1	Hash digest register 1

16.6.2.6 HASH_H2

0x0014			Hash digest register 2											HASH_H2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DR2															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR2															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-10 Hash Digest Register 2 Description

Field	Name	Description
31:0	DR2	Hash digest register 2

16.6.2.7 HASH_H3

0x0018			Hash digest register 3											HASH_H3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DR3															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR3															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-11 Hash Digest Register 3 Description

Field	Name	Description
31:0	DR3	Hash digest register 3

16.6.2.8 HASH_H4

0x001c			Hash digest register 4											HASH_H4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DR4															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR4															
Type	RO															
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-12 Hash Digest Register 4 Description

Field	Name	Description
31:0	DR4	Hash digest register 4

16.6.2.9 HASH_H5

0x0020			Hash digest register 5											HASH_H5		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DR5															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR5															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-13 Hash Digest Register 5 Description

Field	Name	Description
31:0	DR5	Hash digest register 5

16.6.2.10 HASH_H6

0x0024			Hash digest register 6											HASH_H6		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DR6															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR6															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-14 Hash Digest Register 6 Description

Field	Name	Description
31:0	DR6	Hash digest register 6

16.6.2.11 HASH_H7

0x0028			Hash digest register 7											HASH_H7		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DR7															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DR7															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-15 Hash Digest Register 7 Description

Field	Name	Description
31:0	DR7	Hash digest register 7

16.6.2.12 HASH_IMR

0x002c			Hash interrupt enable register											HASH_IMR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														DIGCALIE	DATAINIE
Type	RO														RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-16 Hash Interrupt Enable Register Description

Field	Name	Description
31:2	Reserved	Reserved
1	DIGCALIE	Digest calculation completion interrupt enable 0x0: Disable digest calculation completion interrupt. 0x1: Enable digest calculation completion interrupt.
0	DATAINIE	Data input interrupt enable 0x0: Disable data input interrupt 0x1: Enable data input interrupt

16.6.2.13 HASH_SR

0x0030			Hash status register											HASH_SR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							WORDNUM				DINNE	BUSY	Reserved	DIGCALSTAT	DATAINSTAT
Type	RO							RO				RO	RO	RO	RC_W0	RC_W0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 16-17 Hash Status Register Description

Field	Name	Description
31:9	Reserved	Reserved
8:5	WORDNUM	Number of words already pushed The bit reflects the number of words in the message that has been pushed into FIFO. If DINNE = 1, perform write access to DIN, wordnum is incremented. When init writes 1 or digest calculation starts, wordnum will become 0. 0x0: If DINNE = 0, it means that no word is pushed into the DIN, and both DIN and FIFO are empty. If DINNE = 1, it means that one word has been pushed into DIN. The DIN has one word, but the FIFO is empty. 0x1: 2 words have been pushed into DIN. It means that DIN and FIFO have one word each. ... 0xF: 16 words have been pushed into DIN.
4	DINNE	DIN not empty

Field	Name	Description
3	BUSY	Busy status
2	Reserved	Reserved
1	DIGCALSTAT	Digest calculation completion interrupt status
0	DATAINSTAT	Data input interrupt status

16.6.2.14 HASH_CR1

0x0034			Hash control register 1											HASH_CR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DMAPADEN	DMADINLEN														
Type	RW	RW														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DMADINLEN															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-18 Hash Control Register 1 Description

Field	Name	Description
31	DMAPADEN	Enable auto-populate of message When DMAPADEN = 1, there is no need to configure the final digital calculation. This bit is valid when DMA is enabled.
30:0	DMADINLEN	The length of words transferred by DMA This bit is valid when DMAPADEN = 1.

Advanced-Control Timer (ADVTMR)

This chapter describes the details of the Advanced-Control Timer (ADVTMR).

Topics: **Page**

17.1 Introduction

The Advanced-Control Timers, ADVTMR0, ADVTMR1, and ADVTMR2, consist of a 16-bit auto-reload counter and a programmable prescaler.

The ADVTMR can be used for a variety of purposes, including the measurement of the pulse lengths of input signals (input capture), as well as the generation of output waveforms (output compare, PWM, complementary PWM with dead-time control). The pulse length and period of the waveforms can be configured by using the timer prescaler and the RCC prescalers.

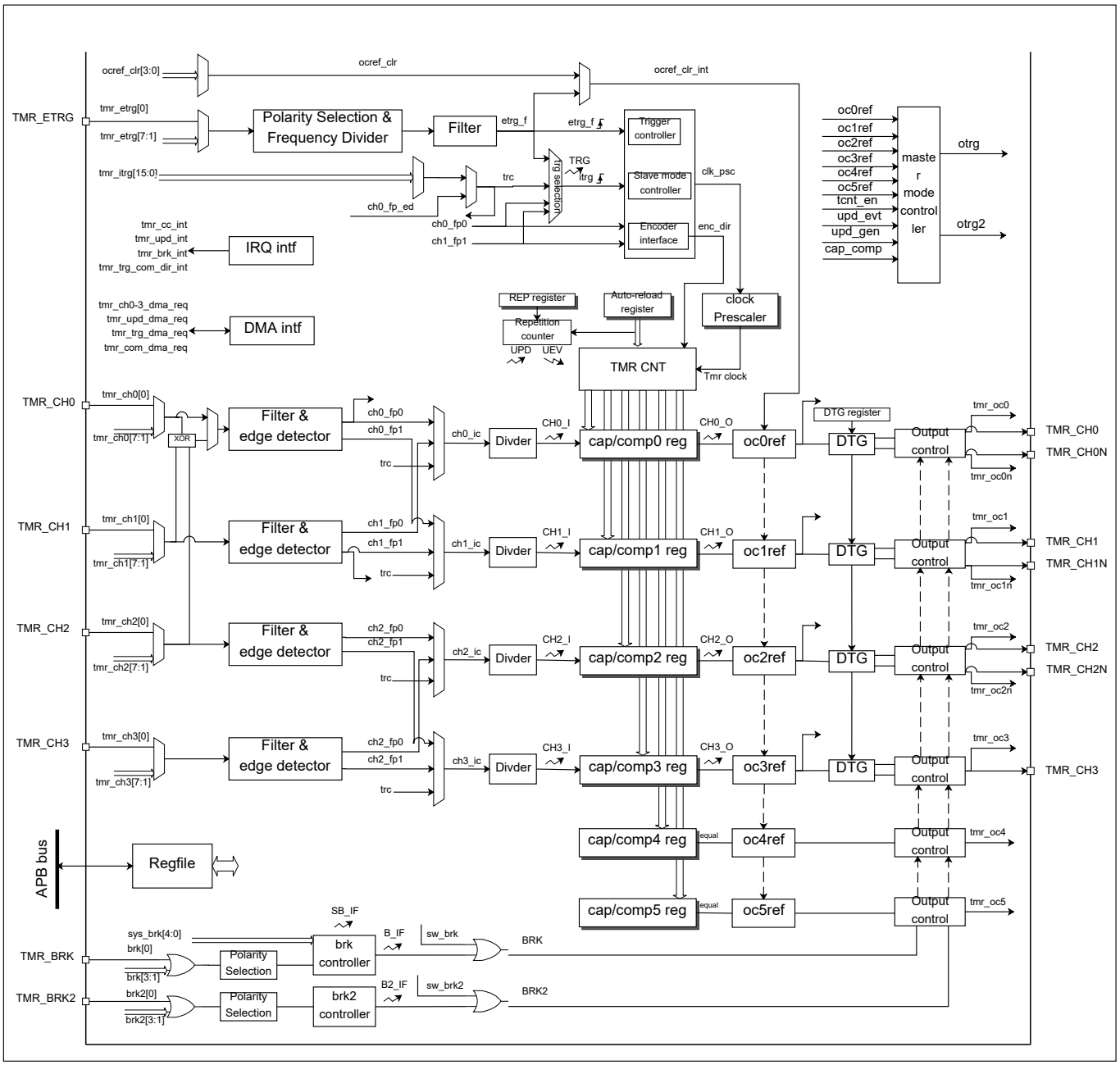
17.2 Features

- 16-bit up/down auto-reload counter
- 16-bit programmable prescaler for dividing the counter clock frequency by any factor between 1 and 65536
- Up to six independent channels (CH0, CH1, CH2, CH3, CH4 and CH5) that can be utilized for the following functions:
  - Input capture (excluding CH4 and CH5)
  - Output compare
  - PWM generation in both edge and center-aligned mode
  - Single-pulse mode output
- Complementary outputs with programmable dead-time control
- Control the timer with external signals or interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Two break input signals to put the timer's output signals in a configurable setting
- Interrupt and DMA generation on the following events:
  - Update event: Counter overflow/underflow, counter initialization with the different internal and external triggers
  - Input capture event
  - Trigger event (counter start, stop, initialization, or count by internal and external triggers)
  - Output compare
- Supports quadrature encoder and hall-sensor circuitry for positioning
- Trigger input for external clock or cycle-by-cycle current management

17.3 Functional Description

17.3.1 Block Diagram

Figure 17-1 illustrates the block diagram of Advanced-Control Timers (ADVTMR0/ADVTMR1/ ADVTMR2).



Notes:

Register — Preload registers transferred to active registers on UEV event according to control bit

Event

Interrupt & DMA requests

Figure 17-1 ADVTMR Block Diagram

17.3.2 ADVTMR0/ADVTMR1/ADVTMR2 Pins and Internal Signals

The following tables provide a summary of the inputs and outputs of the Advanced-Control Timer (ADVTMR).

Table 17-1 ADVTMR Input/Output Pins

Pin Name	Signal Type	Description
TMR_CH0	Input/Output	Timer multi-purpose channels: • Each channel can be used for the capture, comparison or PWM. • TMR_CH0/1 can also be used as an external clock (below 1/4 of the CLK_TMR clock), external trigger, and quadrature encoder inputs. • TMR_CH0/1/2 can be used to interface with digital hall effect sensors.
TMR_CH1	Input/Output	
TMR_CH2	Input/Output	
TMR_CH3	Input/Output	
TMR_CH0N	Output	Timer complementary outputs, which are derived from TMR_CHx outputs with the possibility to have dead-time insertion function.
TMR_CH1N	Output	
TMR_CH2N	Output	
TMR_ETRG	Input	External trigger input. This input can be used as an external trigger or clock source. This input can receive a clock with a frequency higher than the CLK_TMR if the tmr_etrg prescaler is used.
TMR_BRK	Input	Break input
TMR_BRK2	Input	Break input 2

Table 17-2 ADVTMR Internal Input/Output Signals

Internal Signal Name	I/O	Width	Bits	Description
tmr_itrg	Input	[15:0]	16	Internal trigger input bus. These inputs can be used for the slave mode controller or as an input clock (below 1/4 of the CLK_TMR clock).
tmr_etrg	Input	[7:0]	8	External trigger internal input bus.

Internal Signal Name	I/O	Width	Bits	Description
				These inputs can be used as triggers, external clocks, or for hardware cycle-by-cycle pulse width control. These inputs can receive clocks with a frequency higher than the CLK_TMR if the tmr_etrg prescaler is used.
tmr_brk	Input	[3:0]	4	Break input bus for internal signals
tmr_brk2	Input	[3:0]	4	Break input bus 2 for internal signals
tmr_sys_brk	Input	[4:0]	5	System break input. This input gathers MCU's system-level errors.
tmr_ocref_clr	Input	[3:0]	4	Timer reference output clear signals bus. These inputs can be used to clear the timer ocxref signals, typically for hardware cycle-by-cycle pulse width control. They also can be connected to the output of the internal comparators.
tmr_ch0	Input	[7:0]	8	Internal timer inputs bus. These inputs can be used for the capture or as external clocks (below 1/4 of the CLK_TMR clock) and for quadrature encoder signals.
tmr_ch1	Input	[7:0]	8	
tmr_ch2	Input	[7:0]	8	
tmr_ch3	Input	[7:0]	8	
tmr_otrg	Output		1	Internal trigger outputs. These triggers are used by other timers.
tmr_otrg2	Output		1	Internal trigger outputs 2. These triggers are used as the trigger sources for other peripherals, such as ADC.
tmr_cc_int	Output		1	Timer capture/compare interrupt
tmr_upd_int	Output		1	Timer update event interrupt
tmr_brk_int	Output		1	Timer break and break2 interrupt
tmr_trg_com_dir_int	Output		1	Timer trigger, commutation, direction and index interrupt
tmr_ch0_dma_req	Output		1	Timer capture/compare channel 0/1/2/3/4/5 DMA requests
tmr_ch1_dma_req	Output		1	

Internal Signal Name	I/O	Width	Bits	Description
tmr_ch2_dma_req	Output		1	
tmr_ch3_dma_req	Output		1	
tmr_upd_dma_req	Output		1	Timer update DMA request
tmr_trg_dma_req	Output		1	Timer trigger DMA request
tmr_com_dma_req	Output		1	Timer commutation DMA request

Table 17-3 lists the sources connected to the tmr_etrg[7:0] input multiplexers.

Table 17-3 Interconnect to the tmr_etrg[7:0] Input Multiplexer

Timer External Trigger Input Signals	Timer External Trigger Signals Assignment								
	GPTMR0	GPTMR1	GPTMR2	GPTMR3	GPTMR4	GPTMR5	ADVTMR0	ADVTMR1	ADVTMR2
tmrx_etrg[0]	GPTMR0_etrg	GPTMR1_etrg	GPTMR2_etrg	GPTMR3_etrg	GPTMR4_etrg	GPTMR5_etrg	ADVTMR0_etrg	ADVTMR1_etrg	ADVTMR2_etrg
tmrx_etrg[1]	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT
tmrx_etrg[2]	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT
tmrx_etrg[3]	ADC_AWD_OUT	GPTMR0_etrg	GPTMR0_etrg	GPTMR0_etrg	GPTMR1_etrg	GPTMR2_etrg	ADC_AWD_OUT		
tmrx_etrg[4]	ELS	GPTMR2_etrg	GPTMR1_etrg	GPTMR1_etrg	GPTMR2_etrg	GPTMR3_etrg	ELS		
tmrx_etrg[5]		GPTMR3_etrg	GPTMR3_etrg	GPTMR2_etrg	GPTMR3_etrg	GPTMR4_etrg			
tmrx_etrg[6]		GPTMR4_etrg	GPTMR4_etrg	GPTMR4_etrg	GPTMR5_etrg	ADVTMR0_etrg			
tmrx_etrg[7]	Reserved								

Table 17-4 lists the sources connected to the tmr_itrg[15:0] input multiplexers.

Table 17-4 Interconnect to the tmr_itrg[15:0] Input Multiplexer

Timer Internal Trigger Input Signals	Timer External Trigger Signals Assignment								
	GPTMR0	GPTMR1	GPTMR2	GPTMR3	GPTMR4	GPTMR5	ADVTMR0	ADVTMR1	ADVTMR2
tmrx_itrg[0]		GPTMR0_otrg	GPTMR0_otrg	GPTMR0_otrg	GPTMR0_otrg	GPTMR0_otrg	GPTMR0_otrg	GPTMR0_otrg	GPTMR0_otrg
tmrx_itrg[1]	GPTMR1_otrg		GPTMR1_otrg	GPTMR1_otrg	GPTMR1_otrg	GPTMR1_otrg	GPTMR1_otrg	GPTMR1_otrg	GPTMR1_otrg
tmrx_itrg[2]	GPTMR2_otrg	GPTMR2_otrg		GPTMR2_otrg	GPTMR2_otrg	GPTMR2_otrg	GPTMR2_otrg	GPTMR2_otrg	GPTMR2_otrg
tmrx_itrg[3]	GPTMR3_otrg	GPTMR3_otrg	GPTMR3_otrg		GPTMR3_otrg	GPTMR3_otrg	GPTMR3_otrg	GPTMR3_otrg	GPTMR3_otrg
tmrx_itrg[4]	GPTMR4_otrg	GPTMR4_otrg	GPTMR4_otrg	GPTMR4_otrg		GPTMR4_otrg	GPTMR4_otrg	GPTMR4_otrg	GPTMR4_otrg
tmrx_itrg[5]	GPTMR5_otrg	GPTMR5_otrg	GPTMR5_otrg	GPTMR5_otrg	GPTMR5_otrg		GPTMR5_otrg	GPTMR5_otrg	GPTMR5_otrg
tmrx_itrg[6]	ADVTMR0_otrg	ADVTMR0_otrg	ADVTMR0_otrg	ADVTMR0_otrg	ADVTMR0_otrg	ADVTMR0_otrg		ADVTMR0_otrg	ADVTMR0_otrg
tmrx_itrg[7]	ADVTMR1_otrg	ADVTMR1_otrg	ADVTMR1_otrg	ADVTMR1_otrg	ADVTMR1_otrg	ADVTMR1_otrg	ADVTMR1_otrg		ADVTMR1_otrg
tmrx_itrg[8]	ADVTMR2_otrg	ADVTMR2_otrg	ADVTMR2_otrg	ADVTMR2_otrg	ADVTMR2_otrg	ADVTMR2_otrg	ADVTMR2_otrg	ADVTMR2_otrg	
tmrx_itrg[9]	Reserved								
tmrx_itrg[10]									
tmrx_itrg[11]									
tmrx_itrg[12]									
tmrx_itrg[13]									
tmrx_itrg[14]									
tmrx_itrg[15]									

Table 17-5 lists the internal sources connected to the tmr_chx_ocref_clr_i[3:0] input multiplexer.

Table 17-5 Interconnect to tmr_chx_ocref_clr_i[3:0] Input Multiplexer

Timer OCREF Clear Input Signal	Timer OCREF Clear Signal Assignment								
	GPTMR0	GPTMR1	GPTMR2	GPTMR3	GPTMR4	GPTMR5	ADVTMR0	ADVTMR1	ADVTMR2
tmrx_ocref_clear[0]	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT
tmrx_ocref_clear[1]	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT
tmrx_ocref_clear[2]									
tmrx_ocref_clear[3]									

Table 17-6 lists the sources connected to the tmr_ch0[7:0] input multiplexers.

Table 17-6 Interconnect to tmr_ch0[7:0] Input Multiplexer

Timer CH0 Input Signal	Timer ch0 signal assignment / Source								
	GPTMR0	GPTMR1	GPTMR2	GPTMR3	GPTMR4	GPTMR5	ADVTMR0	ADVTMR1	ADVTMR2
tmrx_CH0_in[0]	GPTMR0_CH0	GPTMR1_CH0	GPTMR2_CH0	GPTMR3_CH0	GPTMR4_CH0	GPTMR5_CH0	ADVTMR0_CH0	ADVTMR1_CH0	ADVTMR2_CH0
tmrx_CH0_in[1]	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT
tmrx_CH0_in[2]	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT
tmrx_CH0_in[3]	RTC_Wakeup						ELS		
tmrx_CH0_in[4]	ILS	RTC_Wakeup						ELS	
tmrx_CH0_in[5]	ELS	ILS	RTC_Wakeup						
tmrx_CH0_in[6]	IHS	ELS		RTC_Wakeup		MCO			
tmrx_CH0_in[7]		IHS			RTC_Wakeup				

Table 17-7 lists the sources connected to the tmr_ch1[7:0] input multiplexers.

Table 17-7 Interconnect to tmr_ch1[7:0] Input Multiplexer

Timer CH1 Input Signal	Timer CH1 Signal Assignment / Source								
	GPTMR0	GPTMR1	GPTMR2	GPTMR3	GPTMR4	GPTMR5	ADVTMR0	ADVTMR1	ADVTMR2
tmrx_CH1_in[0]	GPTMR0_CH1	GPTMR1_CH1	GPTMR2_CH1	GPTMR3_CH1	GPTMR4_CH1	GPTMR5_CH1	ADVTMR0_CH1	ADVTMR1_CH1	ADVTMR2_CH1
tmrx_CH1_in[1]	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT			
tmrx_CH1_in[2]	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT			
tmrx_CH1_in[3]						13.56M			13.56M
tmrx_CH1_in[4]									
tmrx_CH1_in[5]									
tmrx_CH1_in[6]									
tmrx_CH1_in[7]									

Table 17-8 lists the sources connected to the tmr_ch2[7:0] input multiplexers.

Table 17-8 Interconnect to tmr_ch2[7:0] Input Multiplexer

Timer CH2 Input Signal	Timer CH2 Signal Assignment / Source								
	GPTMR0	GPTMR1	GPTMR2	GPTMR3	GPTMR4	GPTMR5	ADVTMR0	ADVTMR1	ADVTMR2
tmrx_CH2_in[0]	GPTMR0_CH2	GPTMR1_CH2	GPTMR2_CH2	GPTMR3_CH2	GPTMR4_CH2	GPTMR5_CH2	ADVTMR0_CH2	ADVTMR1_CH2	ADVTMR2_CH2
tmrx_CH2_in[1]	CMP0_OUT	CMP0_OUT	CMP0_OUT						
tmrx_CH2_in[2]	CMP1_OUT	CMP1_OUT	CMP1_OUT						
tmrx_CH2_in[3]						13.56M			13.56M
tmrx_CH2_in[4]									
tmrx_CH2_in[5]									
tmrx_CH2_in[6]									
tmrx_CH2_in[7]									

Table 17-9 lists the sources connected to the tmr_ch3[7:0] input multiplexers.

Table 17-9 Interconnect to tmr_ch3[7:0] Input Multiplexer

Timer CH3 Input Signal	Timer CH3 Signal Assignment / Source								
	GPTMR0	GPTMR1	GPTMR2	GPTMR3	GPTMR4	GPTMR5	ADVTMR0	ADVTMR1	ADVTMR2
tmrx_CH3_in[0]	GPTMR0_CH3	GPTMR1_CH3	GPTMR2_CH3	GPTMR3_CH3	GPTMR4_CH3	GPTMR5_CH3	ADVTMR0_CH3	ADVTMR1_CH3	ADVTMR2_CH3
tmrx_CH3_in[1]				CMP0_OUT	CMP0_OUT	CMP0_OUT			
tmrx_CH3_in[2]				CMP1_OUT	CMP1_OUT	CMP1_OUT			
tmrx_CH3_in[3]									
tmrx_CH3_in[4]									
tmrx_CH3_in[5]									
tmrx_CH3_in[6]									
tmrx_CH3_in[7]									

Table 17-10 lists the sources connected to the tmr_brk1[3:0] input multiplexers.

Table 17-10 Interconnect to tmr_brk1[3:0] Input Multiplexer

tmr_brk1 Input Signal	Timer Break1 Signal Assignment/Source								
	GPTMR0	GPTMR1	GPTMR2	GPTMR3	GPTMR4	GPTMR5	ADVTMR0	ADVTMR1	ADVTMR2
tmrx_brk_i[0]	GPTMR0_BRK_IN pin	GPTMR1_BRK_IN pin	GPTMR2_BRK_IN pin	GPTMR3_BRK_IN pin	GPTMR4_BRK_IN pin	GPTMR5_BRK_IN pin	ADVTMR0_BRK_IN pin	ADVTMR1_BRK_IN pin	ADVTMR2_BRK_IN pin
tmrx_brk_i[1]	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT	CMP0_OUT
tmrx_brk_i[2]	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT	CMP1_OUT
tmrx_brk_i[3]									

Table 17-11 lists the sources connected to the tmr_brk2[3:0] input multiplexers.

Table 17-11 Interconnect to tmr_brk2[3:0] Input Multiplexer

tmr_brk2 Input Signal	Timer break2 Signal Assignment / Source								
	GPTMR 0	GPTMR 1	GPTMR 2	GPTMR 3	GPTMR 4	GPTMR 5	ADVTMR0	ADVTMR1	ADVTMR2
tmrx_brk2_i[0]							ADVTMR0_BRK2_IN pin	ADVTMR1_BRK2_IN pin	ADVTMR2_BRK2_IN pin
tmrx_brk2_i[1]							CMP0_OUT	CMP0_OUT	CMP0_OUT
tmrx_brk2_i[2]							CMP1_OUT	CMP1_OUT	CMP1_OUT
tmrx_brk2_i[3]									

Table 17-12 lists the sources connected to the tmr_sys_brk[3:0] input multiplexers.

Table 17-12 Interconnect to tmr_sys_brk[3:0] Input Multiplexer

System Error Signal Source (tmr_sys_brk)	System Break Interconnect								
	GPTMR0	GPTMR1	GPTMR2	GPTMR3	GPTMR4	GPTMR5	ADVTMR0	ADVTMR1	ADVTMR2
Cortex-STAR core lockup (hard fault) output (tmrx_sys_brk0)	GPTMR0_BRK	GPTMR1_BRK	GPTMR2_BRK	GPTMR3_BRK	GPTMR4_BRK	GPTMR5_BRK	ADVTMR0_BRK	ADVTMR1_BRK	ADVTMR2_BRK
SVS output (tmrx_sys_brk2)									
Clock security system (CSS) (tmrx_sys_brk3)									

17.3.3 Time-Base Unit

The Advanced-Control Timer (ADVTMR) is a 16-bit counter that is accompanied by an auto-reload register. This counter can count up, count down, or perform both up and down counting. Additionally, the counter clock can be divided by a prescaler register. It is worth noting that the counter, the auto-reload register, and the prescaler register can be written or read by software, even while the counter is actively running.

The time-based unit consists of the counter register (TCNT), auto-reload register (ARR), prescaler register (PSC), and repetition counter register (RCR).

ARR is the value that will be preloaded in the actual auto-reload register. When writing to or reading from the auto-reload register, the preload register is accessed. The content of the preload register is immediately transferred into the shadow register or at each update event (UEV), depending on the auto-reload register shadow enable bit (ARRSHDWEN) in the Timer CTRL1 register. The UEV is triggered when the UPDDIS bit in the Timer CTRL1 register equals 0, and the counter overflows with the up-counting setting (or underflow with the down-counting setting). It can also be generated by the software. The generation of the UEV is described in detail for each event generation.

The counter is clocked by the prescaler output CLK_TCNT, which is enabled only when the counter enable bit (TCNTEN) in the Timer CTRL1 register is set.

NOTE: The counter begins counting one clock cycle after the counter enable bit (TCNTEN) is set in the Timer CTRL1 register.

17.3.3.1 Prescaler

The prescaler divides the counter clock frequency by a factor ranging from 1 to 65536. It operates through a 16-bit counter controlled by the Timer PSC register. The prescaler ratio can be modified dynamically as the control register is buffered. The updated prescaler ratio takes effect at the next update event. Figure 17-2 provides an example of how the counter behaves when the prescaler ratio is changed dynamically.

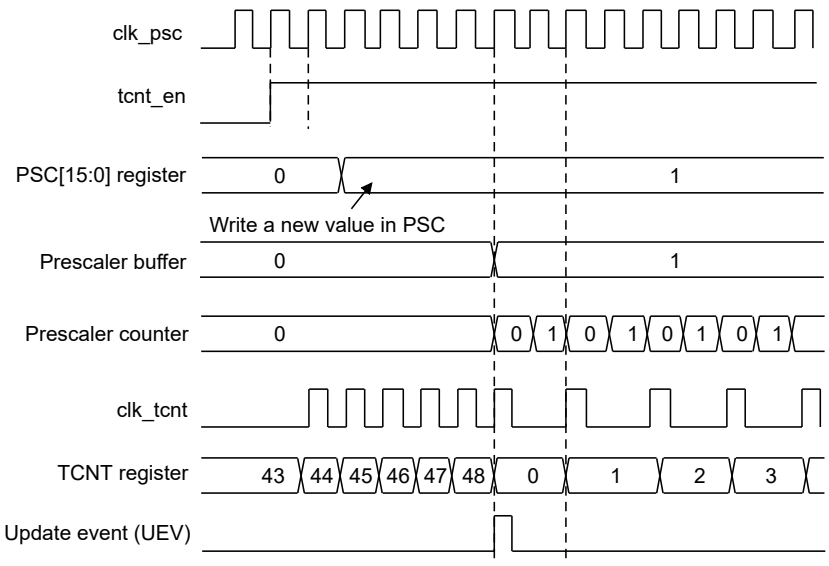


Figure 17-2 Counter Timing Diagram with Prescaler Division Change from 1 to 2

17.3.4 Counter Modes

17.3.4.1 Up-Counting

In up-counting mode, the counter starts at 0 and increments until it reaches the auto-reload value (the value in the Timer ARR register). At that point, the counter resets to 0 and triggers a counter overflow event.

If the repetition counter is enabled, the update event (UEV) will occur after the up-counting process is repeated for the programmed number of times in the repetition counter register (RCR) plus one. Otherwise, the UEV is generated at each counter overflow. Furthermore, enabling the SWUPDGEN bit in the SWEVTGEN register, either through software or by utilizing the slave mode controller, will also result in the generation of a UEV.

To prevent the shadow registers from being updated while writing new values in the preload registers, the software can disable the UEV by setting the update disable bit (UPDDIS) to 1 in the Timer CTRL1 register. Once the UPDDIS bit is set to 1, no update event will occur until the UPDDIS bit is set back to 0. However, this action will cause the counter and the prescaler counter to restart from 0, while the prescale rate remains unchanged.

Additionally, if the update request source (UPDREQSRC) bit in the Timer CTRL1 register is set, enabling the SWUPDGEN bit will generate a UEV without setting the update interrupt flag (UPDIF). As a result, no interrupt or DMA request will be sent. This prevents the generation of both update and capture interrupts when clearing the counter on a capture event.

When a UEV occurs, all the registers are updated, and the update interrupt flag (UPDIF bit in INTSTS register) is set depending on the UPDREQSRC bit:

- The repetition counter is reloaded with the content of Timer RCR register.
- The auto-reload shadow register is updated with the preload value (the content of the Timer ARR register).
- The buffer of the prescaler is reloaded with the preload value (value of the Timer PSC register).

The following figures illustrate various scenarios of the counter's behavior at different clock frequencies, with the register ARR set to 0x36.

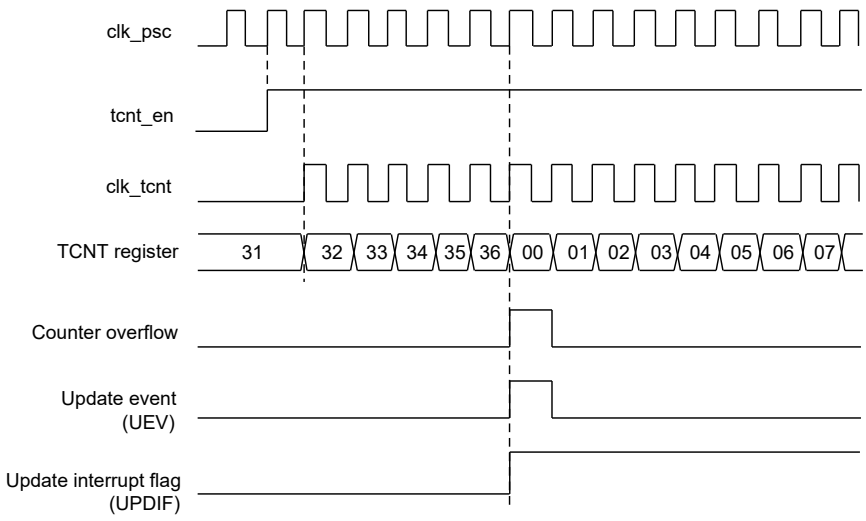


Figure 17-3 Counter Timing Diagram, Internal Clock Divided by 1

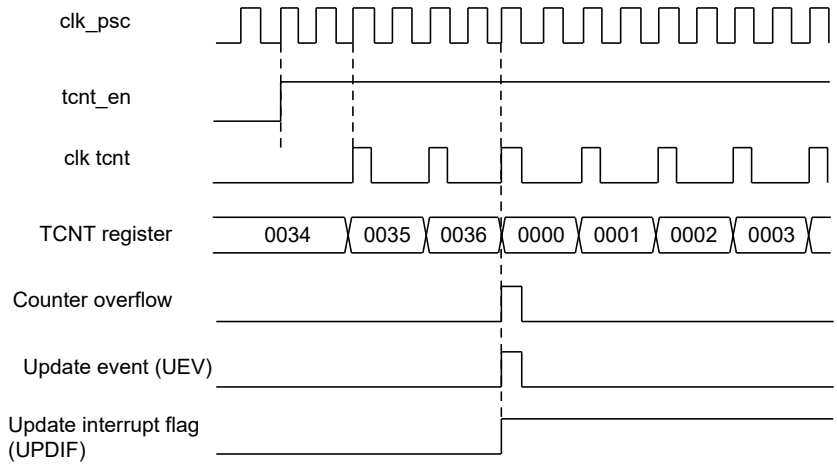


Figure 17-4 Counter Timing Diagram, Internal Clock Divided by 2

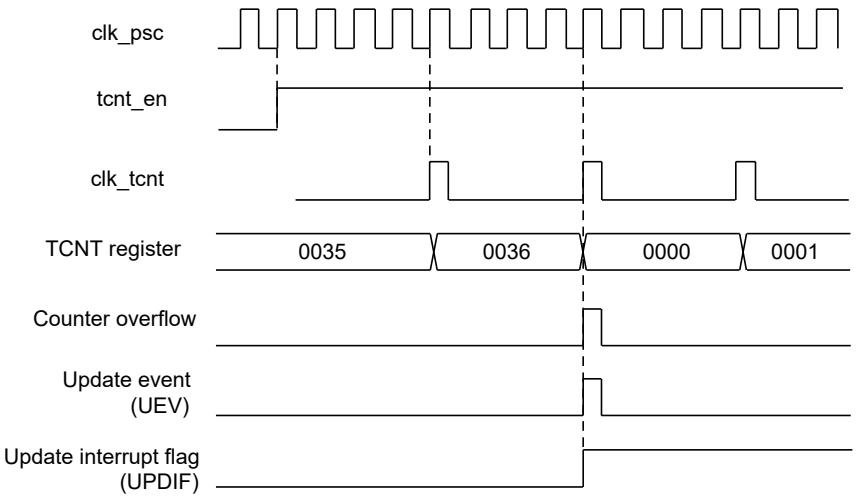


Figure 17-5 Counter Timing Diagram, Internal Clock Divided by 4

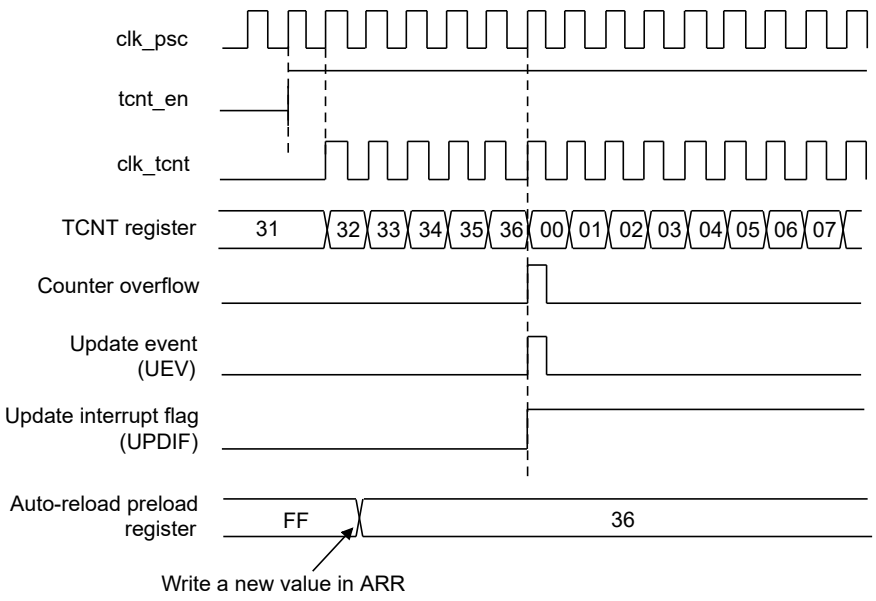


Figure 17-6 Counter Timing Diagram, UEV when ARRSHDWEN = 0 (Timer ARR not Preloaded)

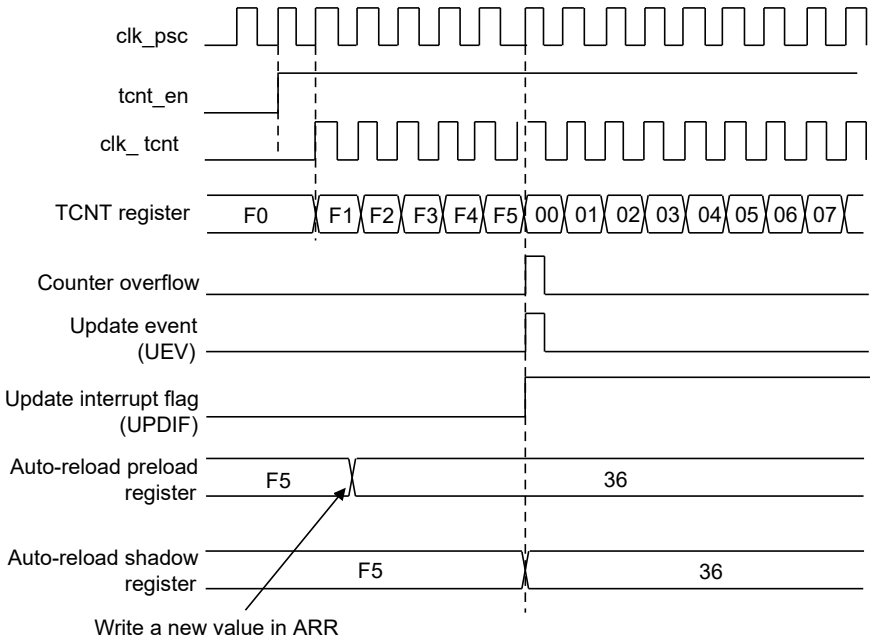


Figure 17-7 Counter Timing Diagram, UEV when ARRSHDWEN = 1 (Timer ARR Preloaded)

17.3.4.2 Down-Counting

In the down-counting mode, the counter starts from the auto-reload value (the value in the Timer ARR register) and counts down to 0. After reaching 0, it restarts from the auto-reload value and triggers a counter underflow event.

If the repetition counter is enabled, the update event (UEV) occurs after the down-counting is repeated for the programmed number of times in the repetition counter register (ADVTMR_RCR) plus one. Otherwise, the UEV occurs at each counter underflow. Additionally, enabling the

SWUPDGEN bit in the SWEVTGEN register, either through software or using the slave mode controller, also triggers an update event.

The software can disable the update event (UEV) by setting the UPDDIS bit in the Timer CTRL1 register. This prevents the shadow registers from being updated while writing new values to the preload registers. Once the UPDDIS bit is set to 0, the UEV will resume. However, it's important to note that while the counter restarts from the current auto-reload value, the counter of the prescaler restarts from 0 (without changing the prescale rate).

Furthermore, when the update request source (UPDREQSRC) bit in the Timer CTRL1 register is enabled, setting the SWUPDGEN bit triggers a UEV without setting the UPDIF flag. As a result, no interrupt or DMA request is generated. This is done to prevent the occurrence of both update and capture interrupts when clearing the counter during a capture event.

When a UEV occurs, all the registers are updated, and the update flag (UPDIF bit in INTSTS register) is set depending on the UPDREQSRC bit:

- The repetition counter is reloaded with the content of the Timer RCR register.
- The buffer of the prescaler is reloaded with the preload value (the content of the Timer PSC register).
- The auto-reload active register is updated with the preload value (the content of the Timer ARR register).

  The auto-reload is updated prior to the counter being reloaded, ensuring that the next period is as expected.

The following figures show examples of the counter behavior for different clock frequencies when Timer ARR = 0x36.

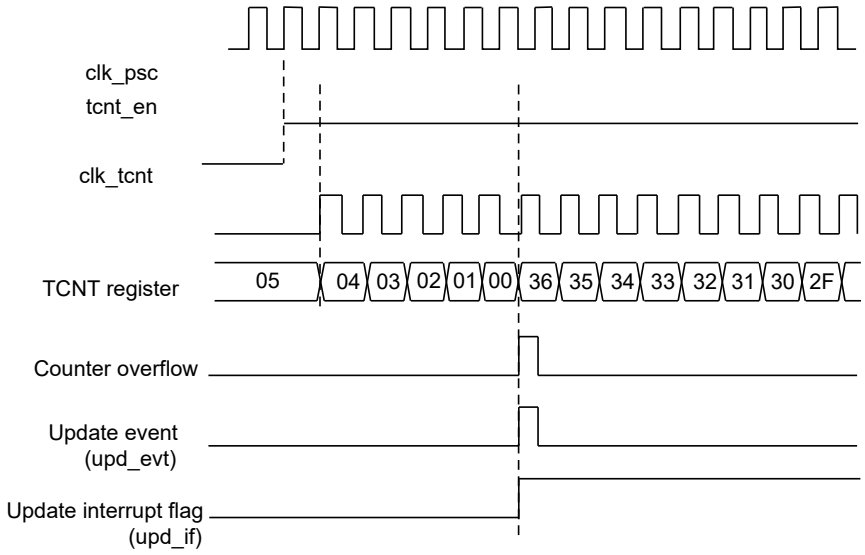


Figure 17-8 Counter Timing Diagram, Internal Clock Divided by 1

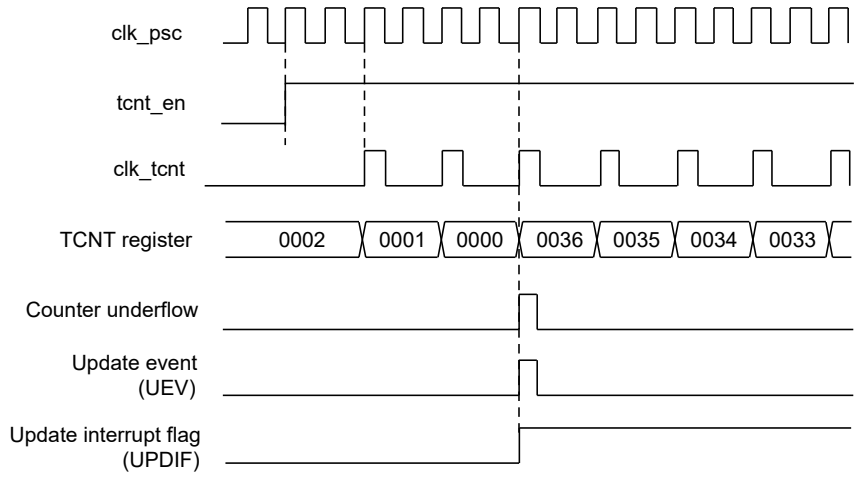


Figure 17-9 Counter Timing Diagram, Internal Clock Divided by 2

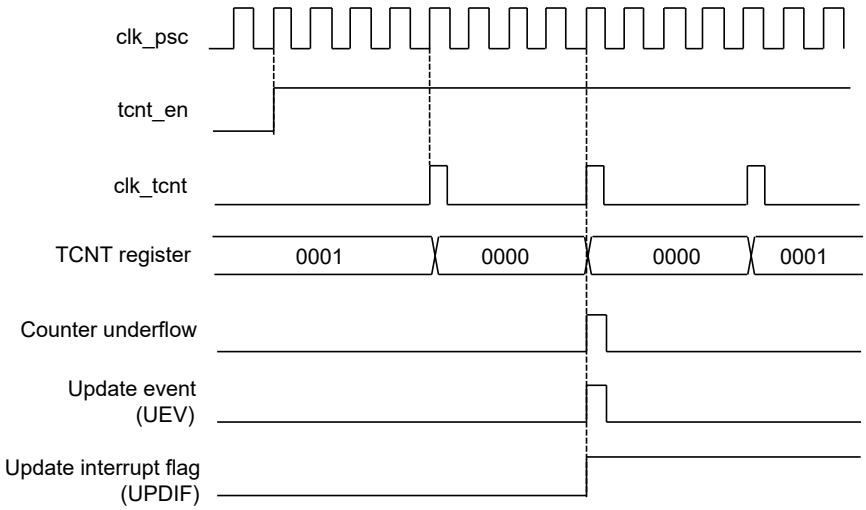


Figure 17-10 Counter Timing Diagram, Internal Clock Divided by 4

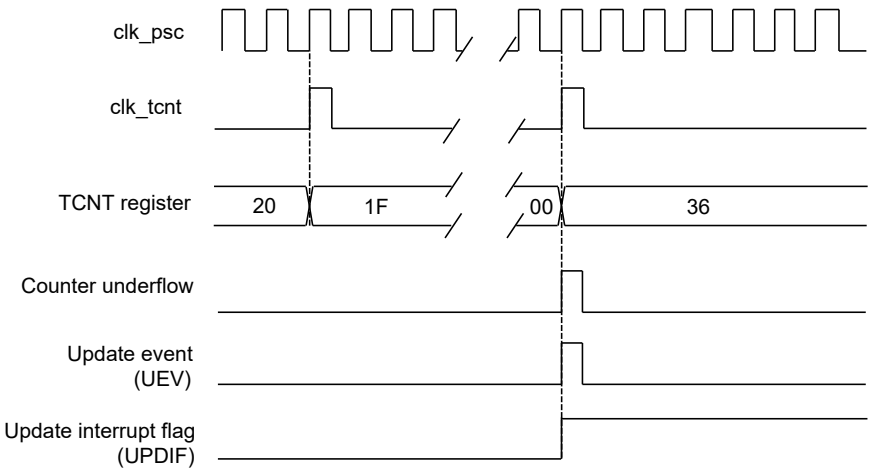


Figure 17-11 Counter Timing Diagram, Internal Clock Divided by N

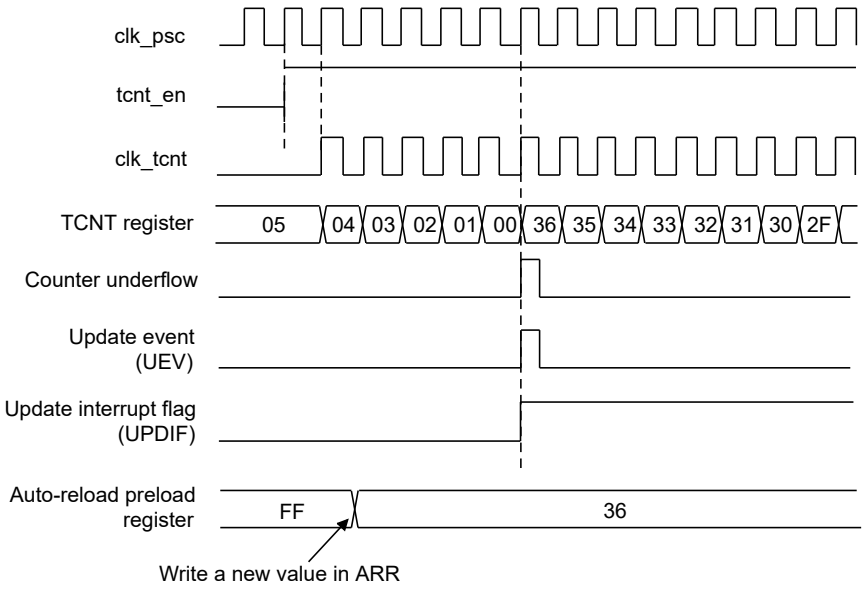


Figure 17-12 Counter Timing Diagram, UEV When Repetition Counter Is not Used

17.3.4.3 Center-Aligned Counting

In the center-aligned counting mode, the counter counts from 0 to the auto-reload value (the content of the Timer ARR register) minus 1. It then generates a counter overflow event. After that, the counter counts from the auto-reload value down to 1 and generates a counter underflow event. Finally, it restarts counting from 0.

This mode is activated when the TCNTALIGNMODE bits in the Timer CTRL1 register are not equal to 00. The output compare interrupt flag for channels configured in output mode is set when TCNTALIGNMODE is set as follows:

- The counter counts down (center-aligned mode 1, TCNTALIGNMODE = 01).
- The counter counts up (center-aligned mode 2, TCNTALIGNMODE = 10).
- The counter counts up and down (center-aligned mode 3, TCNTALIGNMODE = 11).

In this mode, it is not possible to write the DIR direction bit in the Timer CTRL1 register. The hardware automatically updates this bit to indicate the current direction of the counter. The update event (UEV) can occur at each counter overflow and underflow, or by setting the SWUPDGEN bit in the SWEVTGEN register through software or the slave mode controller. When this happens, both the counter and the prescaler restart counting from 0.

The UEV can be disabled by software by setting the UPDDIS bit in the Timer CTRL1 register. This is done to prevent updating the shadow registers while writing new values in the preload registers. In this case, no update event will occur until the UPDDIS bit is set back to 0. However, the counter continues counting up and down, based on the current auto-reload value.

Additionally, if the UPDREQSRC bit (update request source) in Timer CTRL1 register is set, enabling the SWUPDGEN bit will generate a UEV without setting the UPDIF flag. As a result, 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 (UPDIF bit in INTSTS register) is set depending on the UPDREQSRC bit:

- The repetition counter is reloaded with the content of Timer RCR register.
- The auto-reload shadow register is updated with the preload value (ARR).

  If the source of the update is a counter overflow, the autoreload is updated before the counter is reloaded. This ensures the next period is as expected, with the counter loaded with the new value.
- The buffer of the prescaler is reloaded with the preload value (value of the PSC register).

The counter behavior for different clock frequencies is demonstrated in the following figures.

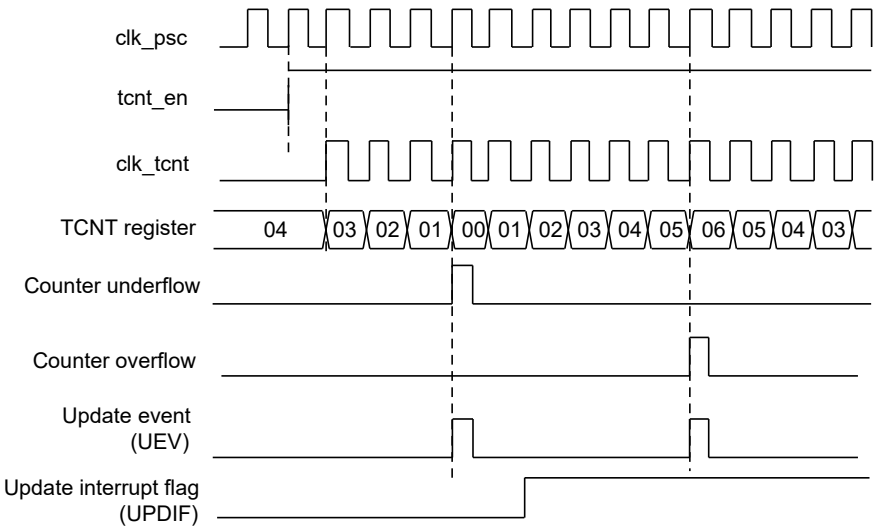


Figure 17-13 Counter Timing Diagram, Internal Clock Divided by 1, Timer ARR = 0x6

NOTE: The center-aligned mode 1 is used.

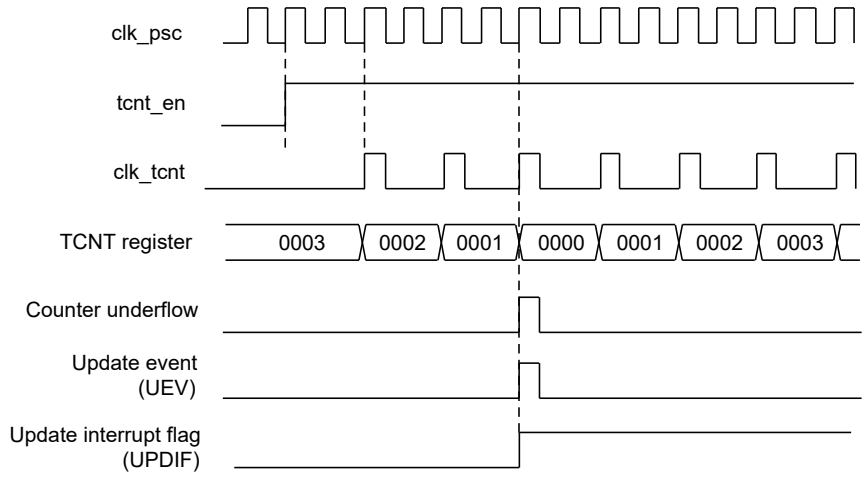


Figure 17-14 Counter Timing Diagram, Internal Clock Divided by 2

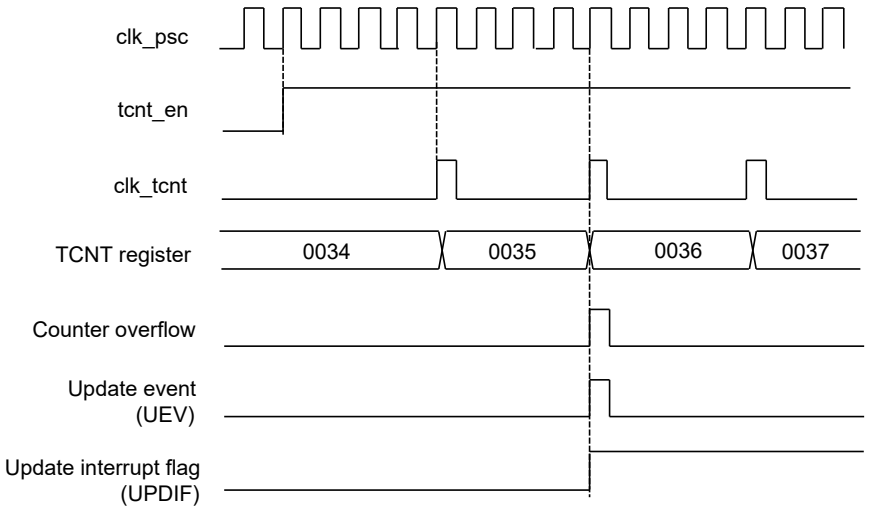


Figure 17-15 Counter Timing Diagram, Internal Clock Divided by 4, Timer ARR = 0x36

NOTE: The center-aligned mode 2 or 3 is updated with an UPDIF on overflow.

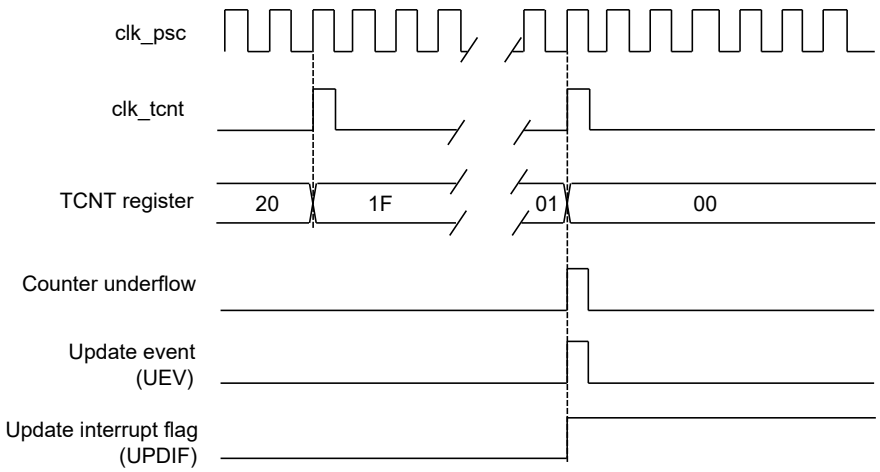


Figure 17-16 Counter Timing Diagram, Internal Clock Divided by N

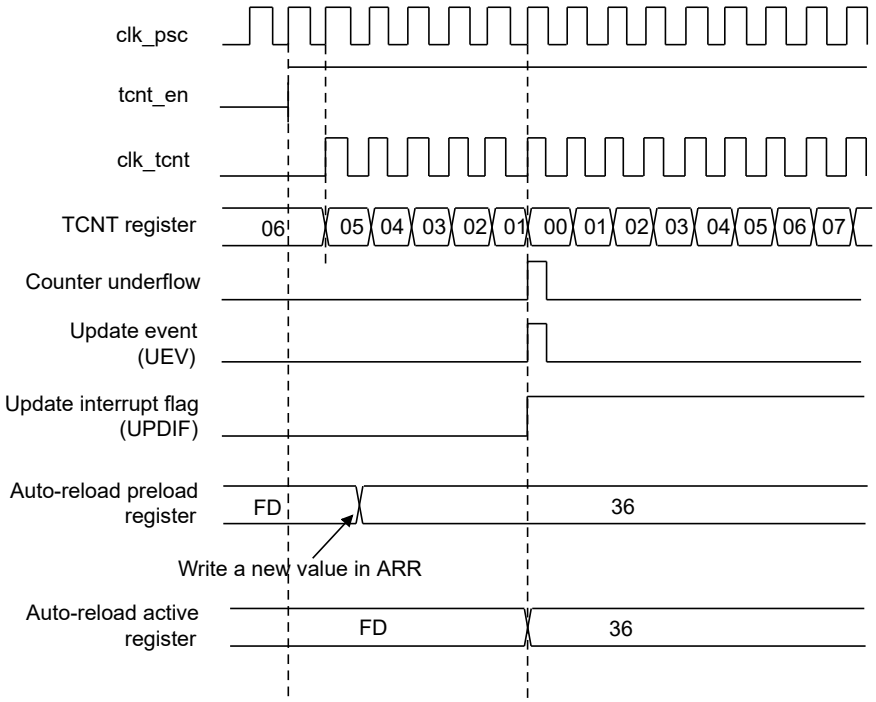


Figure 17-17 Counter Timing Diagram, UEV with ARRSHDWEN = 1 (Counter Underflow)

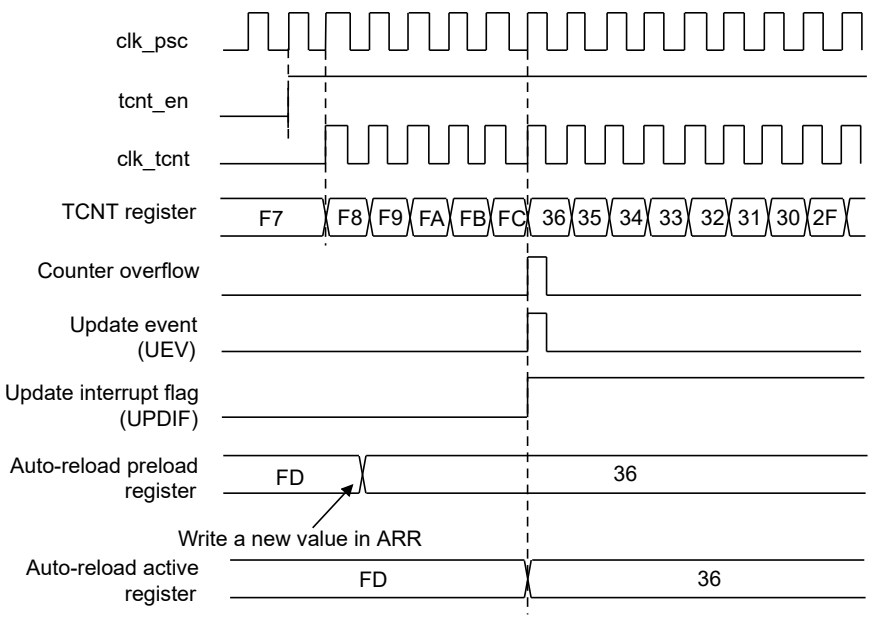


Figure 17-18 Counter Timing Diagram, UEV with ARRSHDWEN = 1 (Counter Overflow)

17.3.5 Repetition Counter

Time-Base Unit explains the generation of the update event (UEV) in relation to the counter overflow or underflow. It is generated only when the repetition counter reaches zero. This feature proves beneficial when generating PWM signals. This implies that the transfer of data occurs from the preload registers to the shadow registers, including the ARR auto-reload register, PSC prescaler register, and Timer CCRx (x = 0, 1, 2, 3) capture/compare registers in compare mode. This transfer happens every N+1 counter overflows or underflows, where N represents the value stored in the Timer RCR repetition counter register.

The repetition counter is reduced in value in the following scenarios:

- During each counter overflow in up-counting mode.
- During each counter underflow in down-counting mode.
- During each counter overflow and underflow in center-aligned mode. This approach restricts the maximum number of repetitions to 32768 PWM cycles, but allows the duty cycle to be updated twice per PWM period.

The repetition counter is of the auto-reload type, and its repetition rate is determined by the value in the Timer RCR register (see Figure 17-19). Whenever the update event is triggered by either the software (by setting the SWUPDGEN bit in the SWEVTGEN register) or the hardware through the slave mode controller, it occurs immediately, regardless of the current value of the repetition counter. Additionally, during the update event, the repetition counter is reloaded with the content of the Timer RCR register.

In the center-aligned counting mode, if the RCR register has an odd value, the update event can occur on either the overflow or underflow. This occurrence is determined by the timing of when the RCR register was written and when the counter was started. If the RCR register was written prior to launching the counter, the update event takes place during the underflow. Conversely, if the RCR register was written after launching the counter, the update event occurs during the overflow.

For instance, if the Timer RCR is set to 3, the UEV is triggered every fourth occurrence of either overflow or underflow, with the timing determined by when the RCR was written.

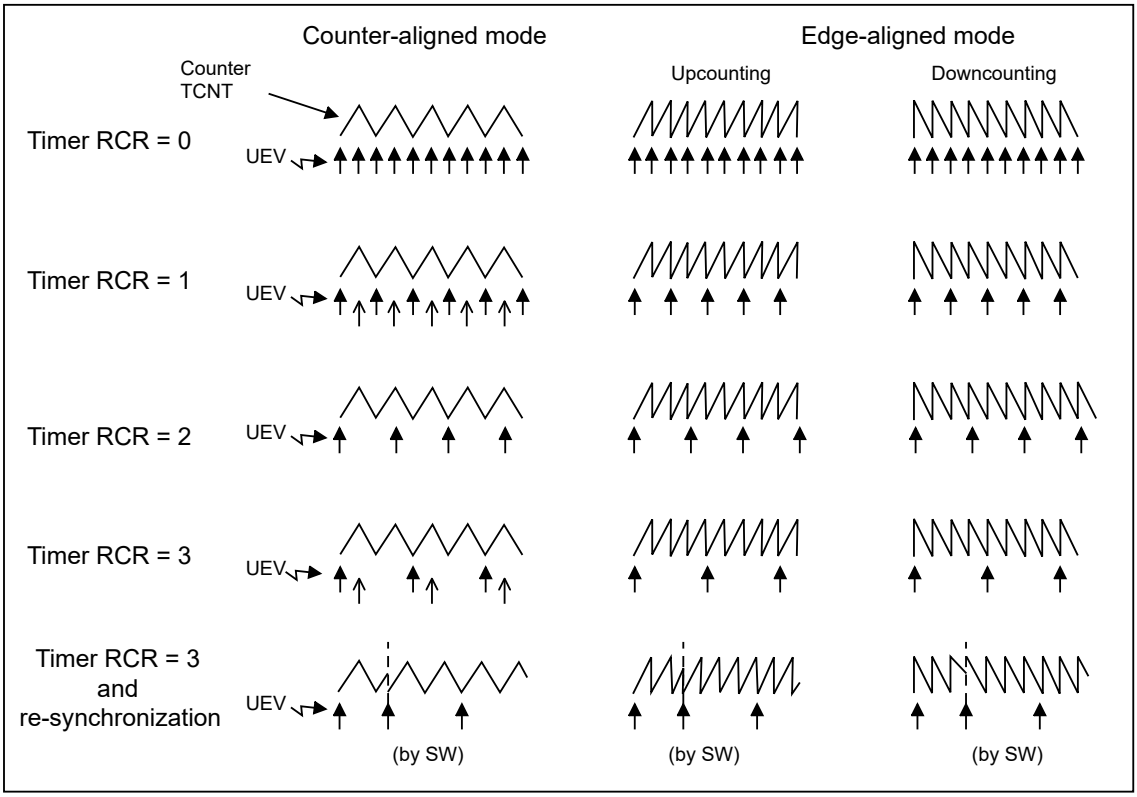


Figure 17-19 Update Rate Examples Depending on Mode and RCR Register Settings

Update event (UEV): Preload registers transferred to active registers and update interrupt generated
Update Event if the repetition counter underflow occurs when the counter is equal to the auto-reload value.

17.3.6 External Trigger Input

The timer features an external trigger input tmr_etrg. It can be used as:

- External clock (external clock mode 2)
- Trigger for the slave mode
- PWM reset input for cycle-by-cycle current regulation

Figure 17-20 describes the tmr_etrg input conditioning. The input polarity is defined with the external trigger polarity ETRGP bit in the SMCFG register. The trigger can be prescaled with the divider programmed by the ETRGFDIVCFG[1:0] bit field and digitally filtered with the ETRGFILTCFG[3:0] bit field. The resulting signal (tmr_etrg) is available for three purposes:

- As an external clock
- Condition the output, typically to reset a PWM output for a current limitation
- As a trigger for the slave mode controller

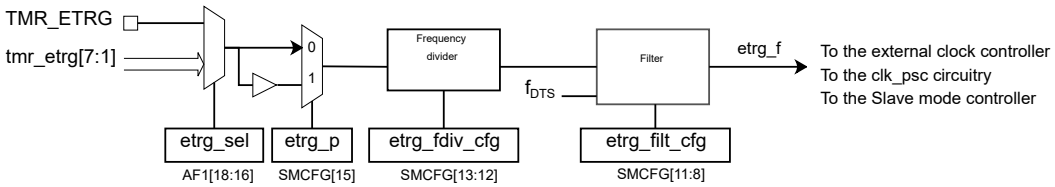


Figure 17-20 External Trigger Input Block

The tmr_etrg input comes from multiple sources:

- Input pins (default configuration)
- Internal sources

The selection is done with the ETRGSEL[3:0] bit field in the Timer AF1 register. Refer to ADVTMR0/ ADVTMR1/ADVTMR2 Pins and Internal Signals for the list of sources connected to the tmr_etrg input in the product.

17.3.7 Clock Selection

The counter clock can be provided by the following clock sources:

- Internal clock (CLK_TMR)
- External clock mode1: external input pin (TMR_CH0 or TMR_CH1)
- External clock mode2: external trigger input (tmr_etrg)
- Encoder mode

17.3.7.1 Internal Clock Source (CLK_TMR)

If the slave mode controller is disabled (SLVMODECTRL = 0000 in the SMCFG regsiter), then the TCNTEN, DIR (in the CTRL1 register) and SWUPDGEN bits (in the SWEVTGEN register) are actual control bits and can be changed only by software (except SWUPDGEN which remains cleared automatically). As soon as the TCNTEN bit is written to 1, the prescaler is clocked by the internal clock CLK_TMR. Figure 17-21 shows the behavior of the control circuit and the up-counter in normal mode, without prescaler.

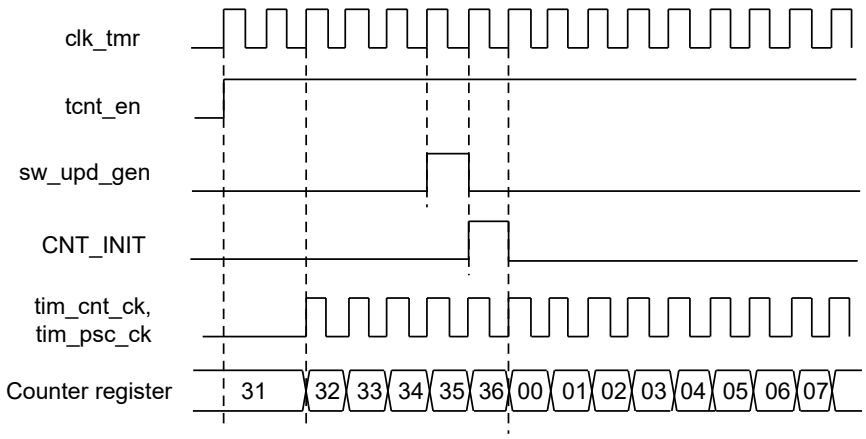


Figure 17-21 Control Circuit in Normal Mode, Internal Clock Divided by 1

17.3.7.2 External Clock Source Mode 0

This mode is selected when SLVMODECTRL = 0111 in the SMCFG register. The counter can count at each rising or falling edge on a selected input.

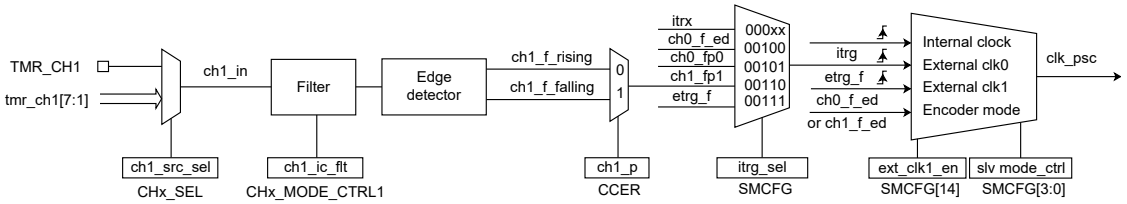


Figure 17-22 tmr_ch1 External Clock Connection Example

For example, to configure the up-counter to count in response to a rising edge on the tmr_ch1 input, follow the steps below:

1. Configure channel 1 to detect rising edges on the tmr_ch1 input by writing CH1MODESEL = '01' in the CHxMODECTRL1 register.
2. Configure the input filter duration by writing the CH1ICFLT[3:0] bits in the CHxMODECTRL1 register (if no filter is needed, keep CH1ICFLT = 0000).
3. Select rising edge polarity by writing CH1P = 0 and CH1NP = 0 in the Timer CCER register.
4. Configure the timer in external clock mode 0 by writing SLVMODECTRL = 0111 in the SMCFG register.
5. Select tmr_ch1 as the trigger input source by writing ITRGSEL = 0110 in the SMCFG register.
6. Enable the counter by writing TCNTEN = 1 in the Timer CTRL1 register.

NOTE: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tmr_ch1, the counter counts once and the TRGIF flag is set. The delay between the rising edge on tmr_ch1 and the actual clock of the counter is due to the resynchronization circuit on tmr_ch1 input.

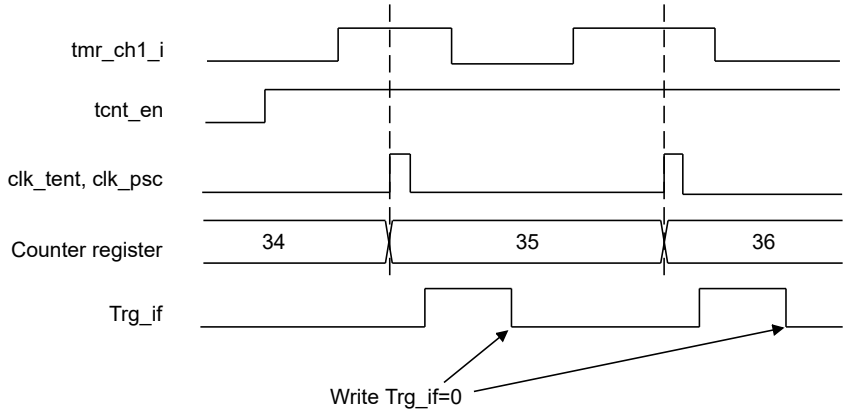


Figure 17-23 Control Circuit in External Clock Mode 0

17.3.7.3 External Clock Source Mode 1

External clock source mode 1 is selected by writing EXTCLKMODE1EN = 1 in the SMCFG register. The counter counts at each rising or falling edge on the external trigger input tmr_etrg. Figure 17-24 gives an overview of the external trigger input block.

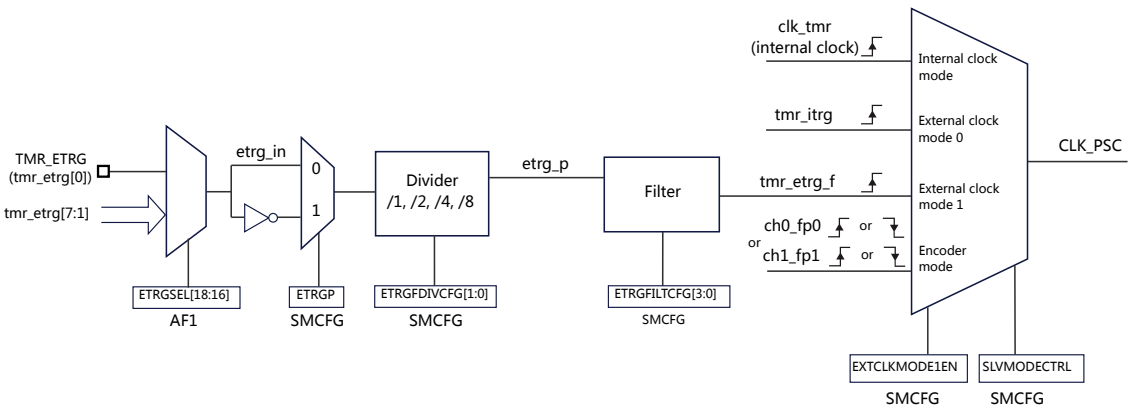


Figure 17-24 External Trigger Input Block

For example, to configure the up-counter to count each 2 rising edges on tmr_etrg, follow the steps below:

1. In the SMCFG register, set ETRGFILTCFG[3:0] to 0000 since no filter is required for this example.
2. Set the prescaler by writing ETRGFDIVCFG[1:0] = 01 in the SMCFG register.
3. Select rising edge detection on the tmr_etrg input by writing ETRGP = 0 in the SMCFG register.
4. Enable external clock mode 2 by writing EXTCLKMODE1EN = 1 in the SMCFG register.
5. Enable the counter by writing TCNTEN = 1 in the Timer CTRL1 register. The counter counts once each of two tmr_etrg rising edges.

The delay between the rising edge on tmr_etrg and the actual clock of the counter is due to the resynchronization circuit on the tmr_etrg_p signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of CLK_TMR frequency. When the tmr_etrg_p signal is faster, the user should apply a division of the external signal by a proper ETRGFDIVCFG prescaler setting.

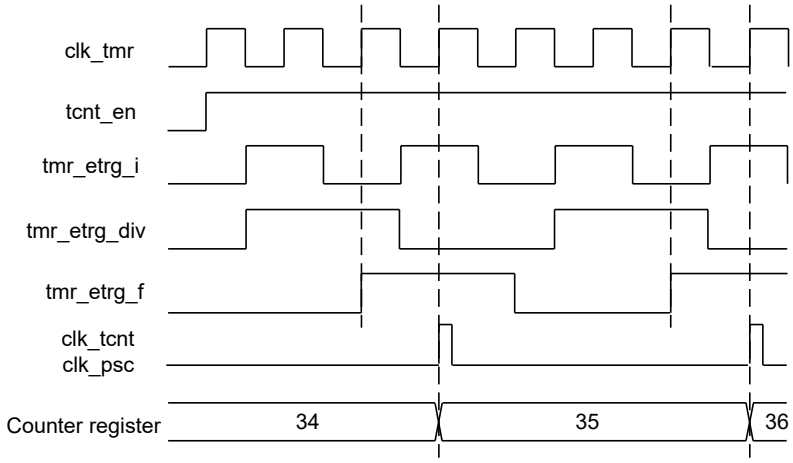


Figure 17-25 Control Circuit in External Clock Mode 1

17.3.8 Capture/Compare Channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with a digital filter, multiplexing, and prescaler, except for channels 4 and 5) and an output stage (with comparator and output control).

Figure 17-26 to Figure 17-27 give an overview of one capture/compare channel.

The input stage samples the corresponding tmr_chx input to generate a filtered signal tmr_chx_f. Then, an edge detector with polarity selection generates a signal (tmr_chx_fp), which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (such as the CH0ICPSC bit in CCMR1[24:0] of ADVTMRx Capture/Compare Mode Register).

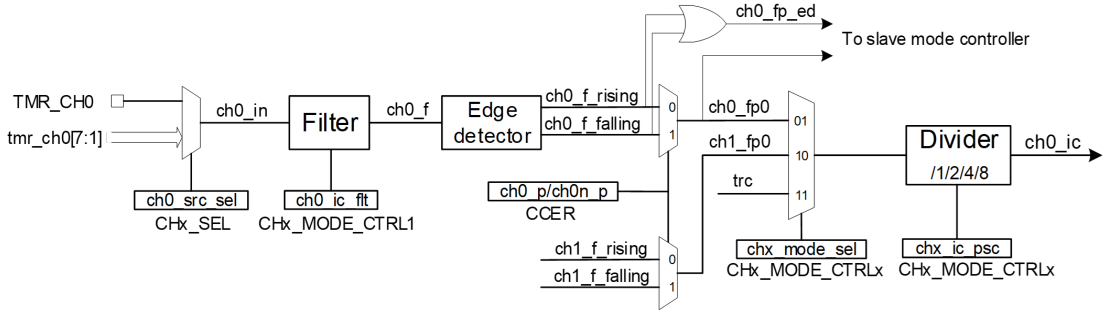


Figure 17-26 Capture/Compare Channel (Example: Channel 1 Input Stage)

The output stage generates an intermediate waveform, which is then used for reference: tmr_ocxref (active high). The polarity acts at the end of the chain.

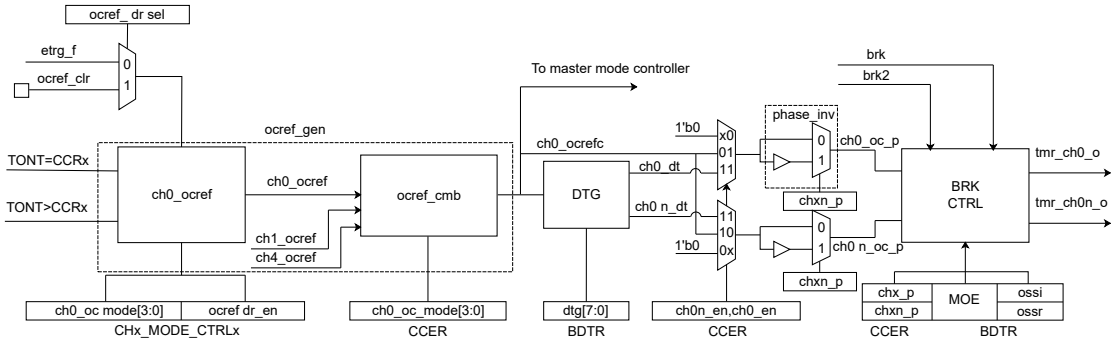


Figure 17-27 Output Stage of Capture/Compare Channel (Channel 0)

(1) tmr_ocxref, where x is the rank of the complementary channel.

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.

17.3.9 Input Capture Mode

In input capture mode, the capture/compare registers (CCRx, x = 0, 1, 2, 3) are used to latch the counter value after a transition detected by the corresponding chx_ic signal. When a capture occurs, the corresponding CHxIF flag (in the INTSTS register) is set, and an interrupt or a DMA request can be sent if enabled. If a capture occurs while the CHxIF flag is already high, then the overcapture flag CHxOF (in the INTSTS register) is set. CHxIF can be cleared by software by writing 0 or reading the captured data stored in the Timer CCRx register. CHxOF is cleared when it is written with 0.

The given example demonstrates how to capture the counter value in Timer CCR0 when the input of tmr_ch0 rises. To achieve this, follow these steps:

1. Select the active input: Connect Timer CCR0 to the tmr_ch0 input by setting the CH0MODESEL bits to 01 in the Timer CHXMODECTRL1 register. When CH0MODESEL is different from 00, the channel is configured for input, and the Timer CCR0 register becomes read-only.

2. Program the appropriate input filter duration in relation to the signal connected to the timer (when the input is one of the tmr_chx (CHxICFLT bits in the Timer CCMRx register).

   For example, when toggling, the input signal is unstable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition

on tmr_ch0 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then, write CH0ICFLT bits to 0011 in the Timer CHXMODECTRL1 register.

3. Select the edge of the active transition on the tmr_ch0 channel by writing CH1P and CH1NP bits to 0 in the Timer CCER register (in this case, the rising edge).
4. Program the input prescaler. In this example, the goal is to capture at each valid transition, so disable the prescaler by setting the CH0ICPSC bits to '00' in the Timer CHXMODECTRL1 register.
5. Enable capture by setting the CH0EN bit in the Timer CCER register.
6. If needed, enable the related interrupt request by setting the CH0IE bit in the Timer DMAINTEN register and/or the DMA request by setting the CH0DE bit in the Timer DMAINTEN register.

When an input capture occurs:

- The Timer CCR0 register gets the counter's value on the active transition.
- CH0IF flag is set (interrupt flag). CH1OF is also set if at least two consecutive captures occurred, whereas the flag was not cleared.
- An interrupt is generated depending on the CH0IE bit.
- A DMA request is generated depending on the CH0DE bit.

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 SWCHxCCGEN (x = 0, 1, 2, 3) bit in the SWEVTGEN register.

17.3.10 PWM Input Mode

This mode allows the measurement of both the period and the duty cycle of a PWM signal connected to single tmr_chx (x = 0, 1, 2, 3) input.

- The Timer CCR0 register holds the period value (interval between two consecutive rising edges).
- The Timer CCR1 register holds the pulse width (interval between two consecutive rising and falling edges.

This mode is a specific instance of the input capture mode. The setup process is similar, but with the following distinctions:

- Two chx_ic signals are mapped on the same tmr_chx_fp1 input.
- These two chx_ic signals are active on edges with opposite polarity.
- One of the two tmr_chx_fpx signals is selected as a trigger input, and the slave mode controller is configured in reset mode.

To measure the period and pulse width of a PWM signal applied to tmr_ch0, follow these steps:

1. Select the active input for Timer CCR0: Set the CH0MODESEL bits to 01 in the Timer CHXMODECTRL1 register (tmr_ch0 selected).
2. Select the active polarity for tmr_ch0fp1 (used for capture in Timer CCR0 and counter clear): Set the CH0P and CH0NP bits to 0 (active on rising edge).
3. Select the active input for Timer CCR1: Set the CH1MODESEL bits to 10 in the Timer CHXMODECTRL1 register (tmr_ch0 selected).

4. Select the active polarity for tmr_ch0fp2 (used for capture in Timer CCR1): Set the CH1P and CH1NP bits to CH1P/CH1NP = 10 (active on falling edge).
5. Select the valid trigger input: Set the ITRGSEL bits to 00101 in the SMCFG register (tmr_ch0fp1 selected).
6. Configure the slave mode controller in reset mode: Set the SLVMODECTRL bits to 0100 in the SMCFG register.
7. Enable the captures: Set the CH0EN and CH1EN bits to 1 in the Timer CCER register.

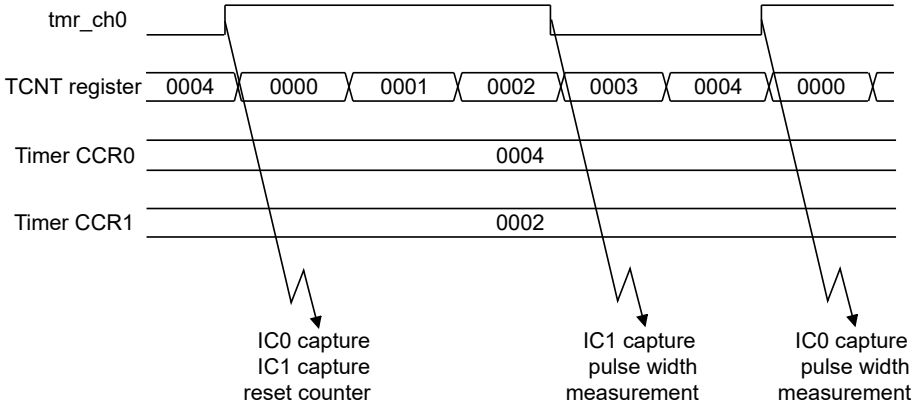


Figure 17-28 PWM Input Mode Timing

17.3.11 Forced Output Mode

In output mode (CHxMODESEL bits = 00 in the Timer CCMRx register), each output compare signal (tmr_ocxref and then tmr_ocx/tmr_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 activate an output compare signal (tmr_ocxref/tmr_ocx), simply set the CHxOCMODE bits in the corresponding Timer CCMRx register to 0101. This will cause tmr_ocxref to be set to a high level (which is always active high), and tmr_ocx will have the opposite value of the CHxP polarity bit. As an example, when CHxP = 0, which means tmr_ocx is active high, tmr_ocx is therefore driven to a high level. The tmr_ocxref signal can be made to go low by writing 0100 to the CHxOCMODE bits in the Timer CCMRx register.

Anyway, the comparison between the Timer CCRx (x = 0, 1, 2, 3) 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.

17.3.12 Output Compare Mode

This mode is designed to manage the output waveform or indicate the elapsed time. Channels 0-3 are available for output, while channels 4 and 5 can only be used within the microcontroller, such as for generating compound waveforms or triggering the ADC.

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 (CHxOCMODE bits in the Timer CCMRx register) and the output polarity (CHxP bit in the Timer CCER register). The output pin can keep its level (CHxOCMODE = 0000), be set active (CHxOCMODE = 0001), be set inactive (CHxOCMODE = 0010) or can toggle (CHxOCMODE = 0011) on match.

- Sets a flag in the interrupt status register (CHxIF bit in the INTSTS register).
- Generates an interrupt if the corresponding interrupt mask is set (CHxIE bit in the Timer DMAINTEN register).
- Sends a DMA request if the corresponding enable bit is set (CHxDE bit in the Timer DMAINTEN register, CCDMAREQSRC bit in the Timer CTRL2 register for the DMA request source selection).

The CCRx registers can be programmed with or without preload registers using the CCSHWDEN bit in the Timer CCMRx register.

In output compare mode, the update event (UEV) does not affect tmr_ocxref and tmr_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in single-pulse mode).

The procedures are as follows.

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the Timer ARR and Timer CCRx registers.
3. Set the CHxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   a. Write CHxOCMODE (x = 0, 1, 2, 3) = 0011 to toggle tmr_ocx output pin when TCNT matches CCRx.
   b. Write CCSHWDEN = 0 to disable the preload/shadow register.
   c. Write CHxP = 0 to select active high polarity.
   d. Write CHxEN = 1 to enable the output.
5. Enable the counter by setting the TCNTEN bit in the Timer CTRL1 register.

The Timer CCRx register can be updated by software at any given time to control the output waveform, as long as the preload/shadow register is not enabled (CCSHWDEN = '0'). If the preload/shadow register is enabled, the Timer CCRx shadow register will only be updated during the UEV event. An example is provided in Figure 17-29.

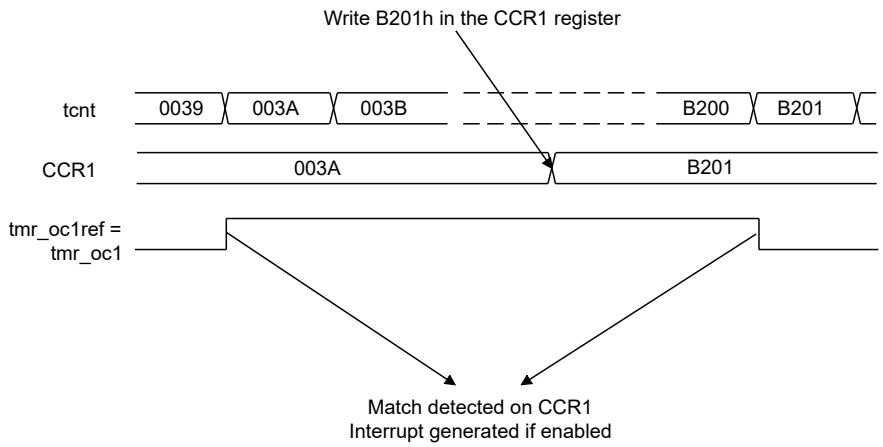


Figure 17-29 Output Compare Mode, Toggle on tmr_oc1

17.3.13 PWM Mode

The Pulse Width Modulation (PWM) mode allows the generation of a signal with a frequency determined by the Timer ARR register value and a duty cycle determined by the Timer CCRx register value.

The PWM mode can be selected independently on each channel (one PWM per tmr_ocx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the CHxOCMODE bits in the Timer CCMRx register. The corresponding preload/shadow register must be enabled by setting the CCSHWDEN bit in the Timer CCMRx register, and eventually, the auto-reload preload register (in up-counting or center-aligned modes) by setting the ARRSHDWEN bit in the Timer CTRL1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the SWUPDGEN bit in the SWEVTGEN register.

tmr_ocx polarity is software programmable using the CHxP bit in the Timer CCER register. It can be programmed as active high or active low. tmr_ocx output is enabled by combining the CHxEN, CHxNEN, MOE, OSSI and OSSR bits (Timer CCER and Timer BDTR registers).

In PWM mode 1 or 2, Timer TCNT and Timer CCRx are always compared to determine whether CCRx ≤ TCNT or TCNT ≤ CCRx (depending on the direction of the counter).

The timer can generate PWM in edge-aligned mode or center-aligned mode depending on the TCNTALIGNMODE bits in the Timer CTRL1 register.

17.3.13.1 PWM Edge-Aligned Mode

Up-Counting Configuration

When the DIR bit in the Timer CTRL1 register is set to low, the up-counting mode is activated. For more information, refer to Up-Counting.

In the specific case of PWM mode 1, the reference PWM signal, tmr_ocxref, remains high as long as TCNT is less than CCRx. However, once TCNT becomes greater than or equal to CCRx, tmr_ocxref transitions to a low state. If the compare value in Timer CCRx exceeds the auto-reload value (stored in Timer ARR), tmr_ocxref is held at '1'. On the other hand, if the compare value is set to 0, tmr_ocxref is held at '0'.

Figure 17-30 shows some edge-aligned PWM waveforms in an example where Timer ARR = 8.

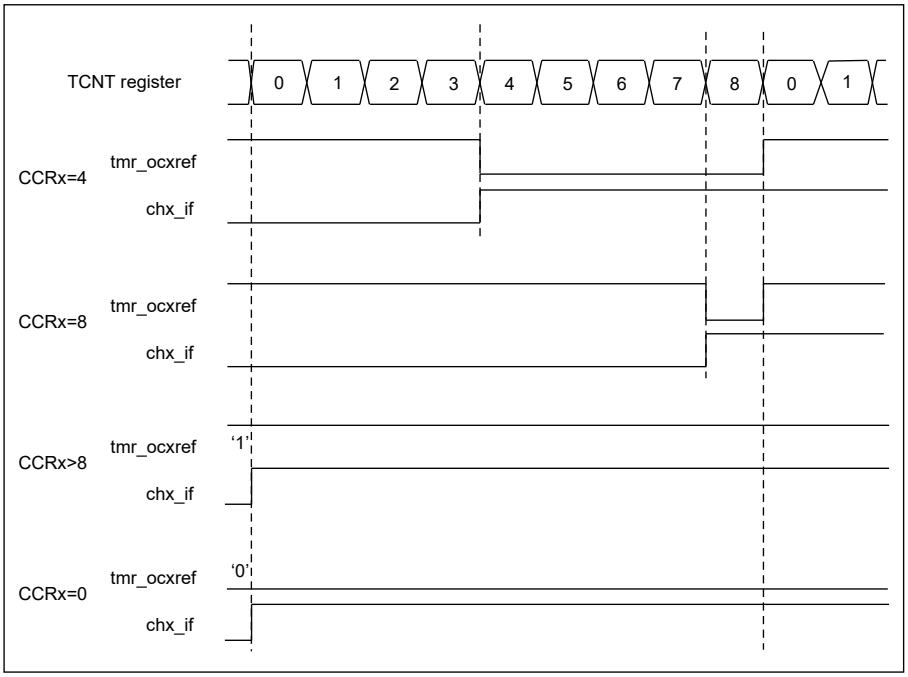


Figure 17-30 Edge-Aligned PWM Waveforms (ARR = 8)

Down-Counting Configuration

When the DIR bit in the Timer CTRL1 register is set to high, the down-counting mode is enabled. For more information, refer to Down-Counting.

In PWM mode 1, the reference signal tmr_ocxref is low as long as TCNT > CCRx else it becomes high. If the compare value in Timer CCRx is greater than the auto-reload value in Timer ARR, then tmr_ocxref is held at '1'. 0% PWM is not possible in this mode.

17.3.13.2 PWM Center-Aligned Mode

The center-aligned mode becomes active when the TCNTALIGNMODE bits in the Timer CTRL1 register are not '00'. All other configurations have the same effect on the tmr_ocxref/tmr_ocx signals. The compare flag is set based on the counter's direction, whether it counts up, down, or both, depending on the TCNTALIGNMODE bits configuration. The direction bit (DIR) in the Timer CTRL1 register is updated by hardware and should not be modified by software.

Figure 17-31 shows some center-aligned PWM waveforms in an example where:

- Timer ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down, corresponding to the selection of center-aligned mode 1 with TCNTALIGNMODE = 01 in the Timer CTRL1 register.

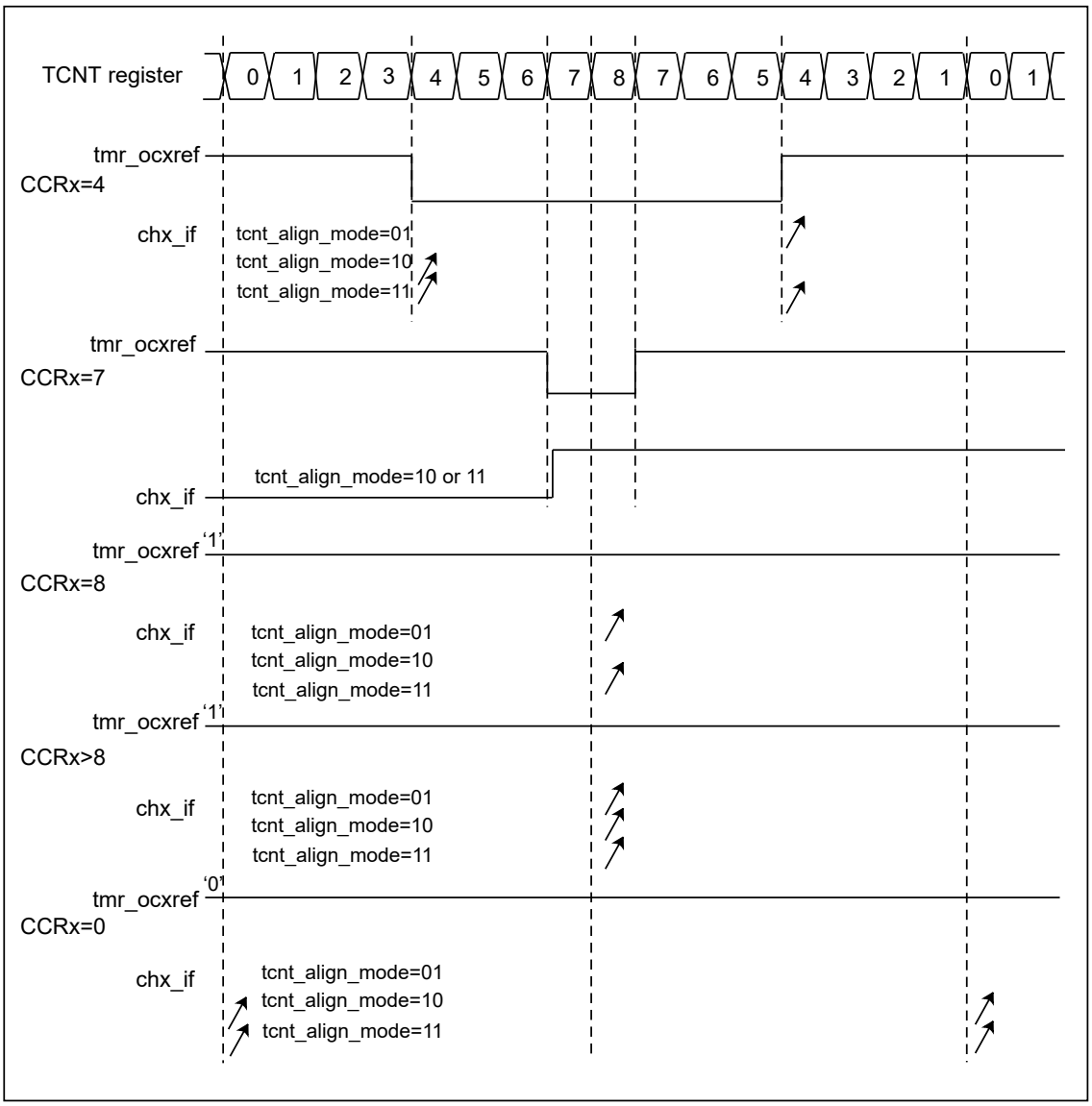


Figure 17-31 Center-Aligned PWM Waveforms (ARR = 8)

Here are some suggestions for 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 based on the value specified in the DIR bit in the Timer CTRL1 register. It is important to note that the software should not modify the DIR and TCNTALIGNMODE bits simultaneously.
- It is not advisable to write to the counter while it is running in center-aligned mode because it can cause unexpected outcomes. Specifically:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TCNT > ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the Timer ARR value is written in the counter, but no update event (UEV) is generated.
- To ensure the safest usage of center-aligned mode, it is recommended to generate an update through software by setting the SWUPDGEN bit in the SWEVTGEN register just before starting the counter. It is also advised not to write to the counter while it is running.

17.3.14 Asymmetric PWM Mode

The asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the Timer ARR register, the duty cycle and the phase shift are determined by a pair of Timer CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tmr_oc0refc (or tmr_oc1refc) is controlled by Timer CCR0 and Timer CCR1
- tmr_oc2refc (or tmr_oc3refc) is controlled by Timer CCR2 and Timer CCR3

This mode can be selected independently on two channels (one tmr_ocx output per pair of CCR registers) by writing 1110 (asymmetric PWM mode 1) or 1111 (asymmetric PWM mode 2) in the CHxOCMODE bits in the Timer CCMRx register.

NOTE: To ensure compatibility, the CHxOCMODE[3:0] bit field is divided into two parts, where the most significant bit is not adjacent to the three least significant bits.

When utilizing a specific channel as an asymmetric PWM channel, it is also possible to use its complementary channel. For example, if a tmr_oc0refc signal is generated on channel 0 in asymmetric PWM mode 1, it is possible to output either the tmr_oc1ref signal on channel 1 or the tmr_oc1refc signal resulting from asymmetric PWM mode 1.

Figure 17-32 represents an example of signals that can be generated using asymmetric PWM mode (channels 0 to 3 are configured in asymmetric PWM mode 2). Together with the dead-time generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

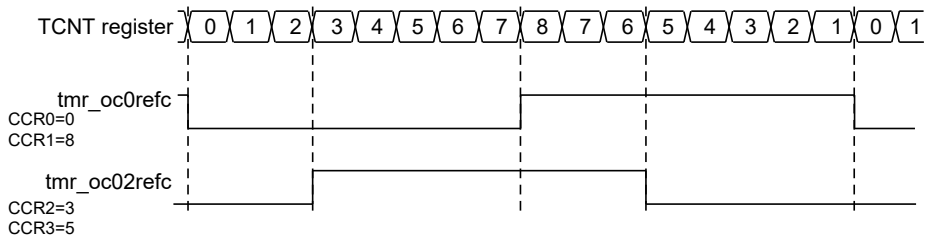


Figure 17-32 Generation of 2 Phase-shifted PWM Signals with 50% Duty Cycle

17.3.15 Combined PWM Mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the Timer ARR register, the duty cycle and delay are determined by the two Timer CCRx registers. The resulting signals, tmr_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tmr_oc0refc (or tmr_oc1refc) is controlled by Timer CCR0 and Timer CCR1
- tmr_oc2refc (or tmr_oc3refc) is controlled by Timer CCR2 and Timer CCR3

Combined PWM mode can be selected independently on two channels (one tmr_ocx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the CHxOCMODE bits in the Timer CCMRx register.

When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in combined PWM mode 1 and the other in combined PWM mode 2).

The CHxOCMODE[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 17-33 illustrates an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 0 is configured in combined PWM mode 2.
- Channel 1 is configured in PWM mode 1.
- Channel 2 is configured in combined PWM mode 2.
- Channel 3 is configured in PWM mode 1.

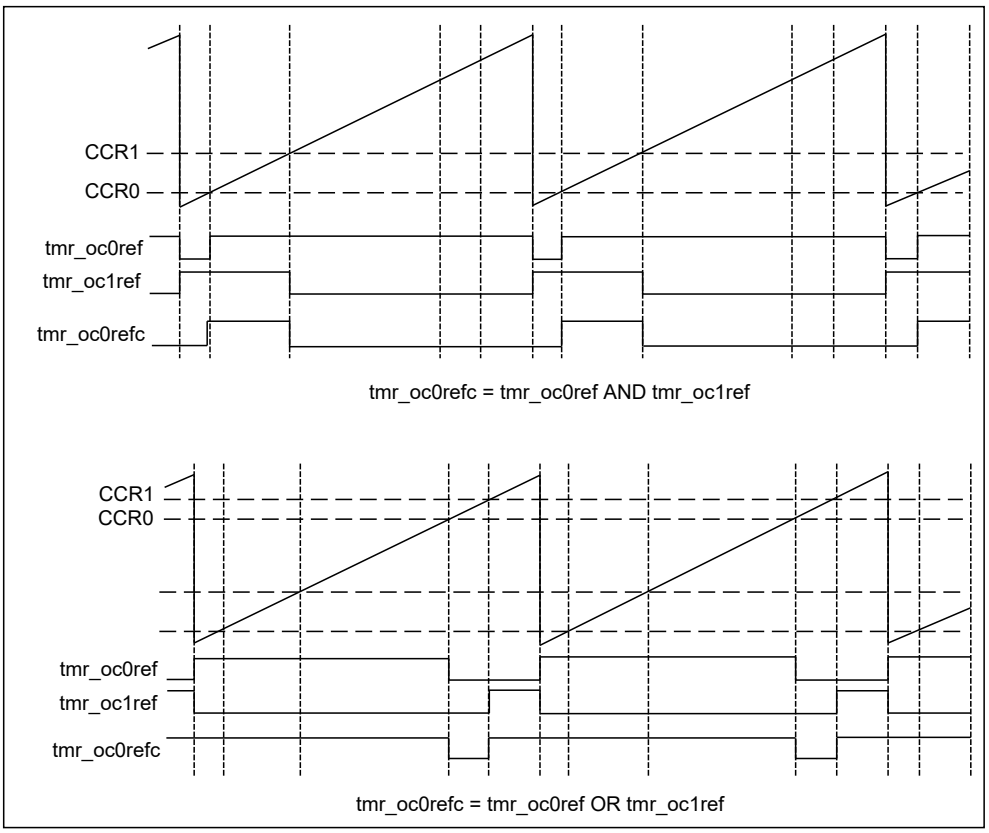


Figure 17-33 Combined PWM Mode on Channel 0 and 2

17.3.16 Combined 3-Phase PWM Mode

The combined 3-phase PWM mode enables the generation of one to three center-aligned PWM signals by using a single programmable signal that is ANDed in the middle of the pulses. The tmr_oc5ref signal is used to define the resulting combined signal. The 3-bit GC4C[3:1] in Timer CCR4 allows for the selection of the reference signal to be combined with tmr_oc4ref. The resulting signals, tmr_ocxrefc, are created by performing an AND logical combination of two reference PWMs:

- If GC4C0 is set, tmr_oc0refc is controlled by Timer CCR0 and CCR4.
- If GC4C1 is set, tmr_oc1refc is controlled by Timer CCR1 and CCR4.
- If GC4C2 is set, tmr_oc2refc is controlled by Timer CCR2 and CCR4.

The combined 3-phase PWM mode can be selected independently on channels 0 to 2 by setting at least one of the 3-bit GC4C[3:1].

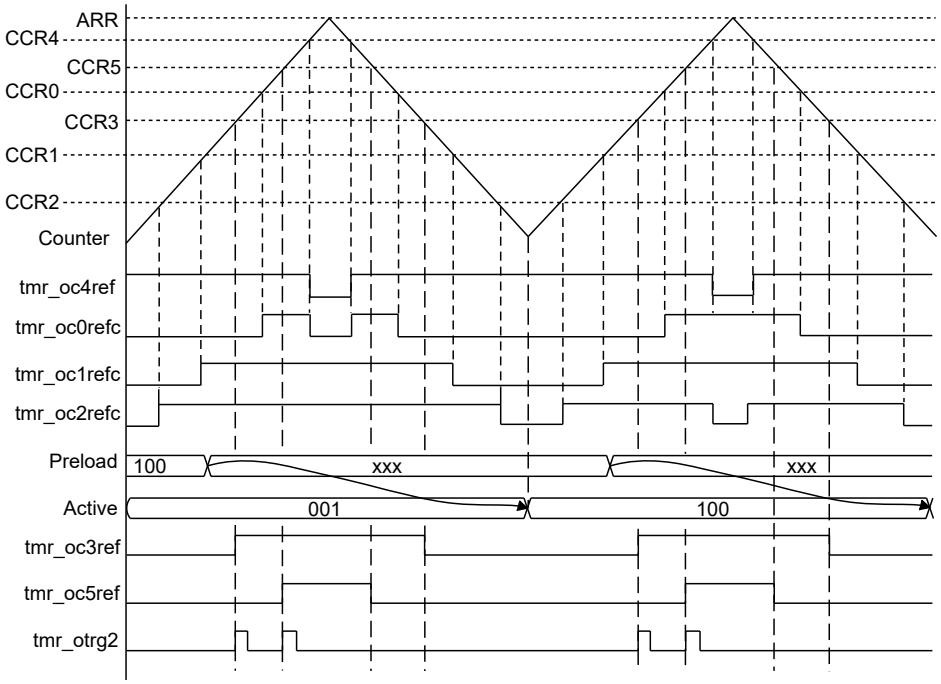


Figure 17-34 3-Phase Combined PWM Signals with Multiple Trigger Pulses per Period

The tmr_otrg2 waveform demonstrates the synchronization of the ADC with 3-phase PWM signals. For more details, refer to ADC Synchronization.

17.3.17 Complementary Outputs and Dead-Time Insertion

The Advanced-Control Timers (ADVTMR0/ADVTMR1/ADVTMR2) 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 adjusted depending on the devices connected to the outputs and their characteristics (intrinsic delays of level- shifters, delays due to power switches).

The polarity of the outputs (main output tmr_ocx or complementary tmr_ocxn) can be selected independently for each output. This is done by writing to the CHxP and CHxNP bits in the Timer CCER register.

The complementary signals tmr_ocx and tmr_ocxn are activated by a combination of several control bits: the CHxEN and CHxNEN bits in the Timer CCER register and the MOE, CHxOIS, CHxNOIS, OSSI and OSSR bits in the Timer BDTR and CTRL2 registers. Refer to output control bits for complementary tmr_ocx and tmr_ocxn channels with the break feature in the register for more details. Specifically, the dead-time is activated when switching to the idle state, indicated by MOE falling to 0.

Dead-time insertion is enabled by setting both CHxEN and CHxNEN bits and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tmr_ocxref, it generates two outputs, tmr_ocx and tmr_ocxn. If tmr_ocx and tmr_ocxn are active high:

- The tmr_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 tmr_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 (tmr_ocx or tmr_ocxn), then the corresponding pulse is not generated.

The following figures illustrate the connections between the output signals of the dead-time generator and the reference signal tmr_ocxref. It is assumed that in these examples, CHxP = 0, CHxNP = 0, MOE = 1, CHxEN = 1 and CHxNEN = 1.

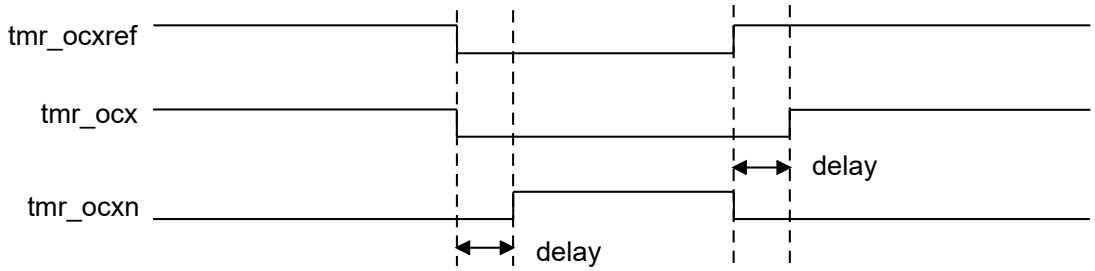


Figure 17-35 Complementary Output with Symmetrical Dead-time Insertion

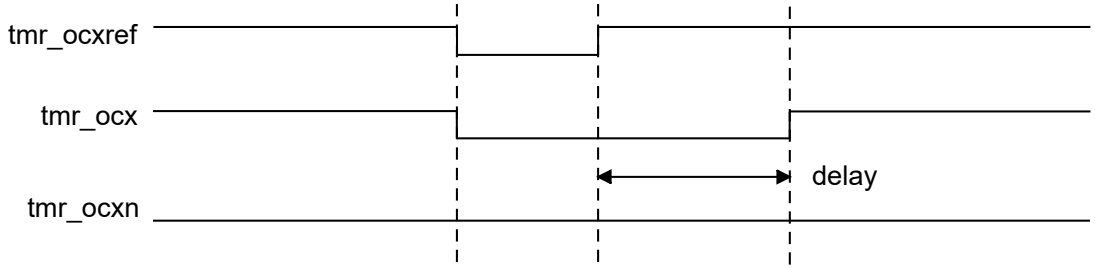


Figure 17-36 Dead-time Waveforms with Delay Greater than the Negative Pulse

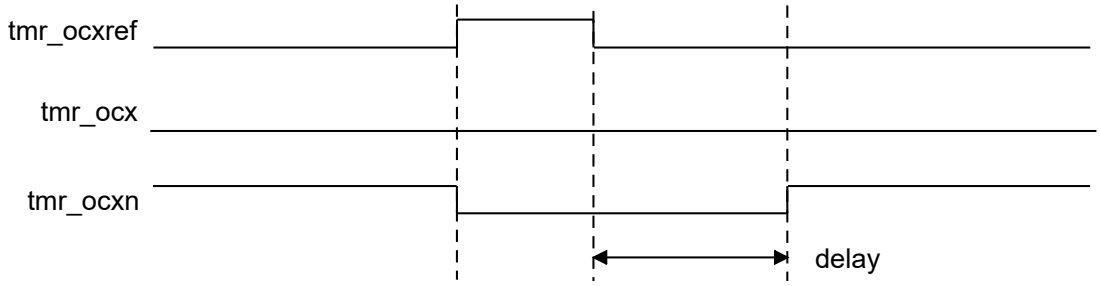


Figure 17-37 Dead-time Waveforms with Delay Greater than the Positive Pulse

The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the Timer BDTR register. Refer to the Timer break and dead-time register (BDTR) for delay calculation.

Re-directing tmr_ocxref to tmr_ocx or tmr_ocxn

In output mode (forced, output compare or PWM), tmr_ocxref can be re-directed to the tmr_ocx output or to tmr_ocxn output by configuring the CHxEN and CHxNEN bits in the Timer CCER register.

This allows sending a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at an inactive level or both outputs active and complementary with dead-time.

NOTE: When only tmr_ocxn is enabled (CHxEN = 0, CHxNEN = 1), it is not complemented and becomes active as soon as tmr_ocxref is high. For example, if CHxNP = 0, then tmr_ocxn= tmr_ocxref.

On the other hand, when both tmr_ocx and tmr_ocxn are enabled (CHxEN = CHxNEN = 1), tmr_ocx becomes active when tmr_ocxref is high, while tmr_ocxn is complemented and becomes active when tmr_ocxref is low.

17.3.18 Break Function

The break function is designed to protect power switches controlled by PWM signals produced by timers. Typically, the two break inputs are linked to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry disables the PWM outputs and directs them to a predetermined safe state Additionally, various internal MCU events can be chosen to trigger an output shutdown.

The break features two channels. A break channel gathers both system-level faults (such as clock failure and ECC/parity errors) and application faults (from input pins and built-in comparator) and can force the outputs to a predefined level (either active or inactive) after a dead-time duration. A break2 channel only includes application faults and can force the outputs to an inactive state.

The output enable signal and output levels during break depend on several control bits:

- The MOE bit in Timer BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the Timer BDTR register defines whether the timer controls the output in an inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The CHxOIS and CHxNOIS bits in the Timer CTRL2 register which are setting the output shut-down level, either active or inactive. The tmr_ocx and tmr_ocxn outputs cannot be set both to active level at a given time, whatever the CHxOIS and CHxNOIS values. Refer to output control bits for complementary tmr_ocx and tmr_ocxn channels with break feature in the registers for more details.

When exiting from reset, the break circuit is disabled, and the MOE bit is low. The break functions can be enabled by setting the BRKEN and BRK2EN bits in the Timer BDTR register. The break input polarities can be selected by configuring the BRKP and BRK2P bits in the same register. BRKxEN and BRKxP can be modified at the same time. When the BRKxEN and BRKxP 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 the 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 Timer BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1, whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal, whereas the read reflects the synchronous signal.

The sources for break (tmr_brk) channel are:

- External sources connected to one of the TMR_BRK and TMR_BRK2 pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:

  - Coming from a tmr_brk input (refer to ADVTMR0/ADVTMR1/ADVTMR2 Pins and Internal Signals for product specific implementation)
  - Coming from a system break request sys_brk (refer to ADVTMR0/ADVTMR1/ADVTMR2 Pins and Internal Signals for product specific implementation)

The sources for break2 (tmr_brk2) are:

- External sources connected to one of the TMR_BRK2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tmr_brk2 input (refer to ADVTMR0/ADVTMR1/ADVTMR2 Pins and Internal Signals for product specific implementation)

Break events can also be generated by software using SWBRKGEN and SWBRK2GEN bits in the SWEVTGEN register.

All sources are ORed before entering the timer tmr_brk or tmr_brk2 inputs, they can be found in the circuitry of Break and Break2.

NOTE: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in an inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the CHxOIS bit in the Timer CTRL2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in an inactive state (depending on the polarity). This is done asynchronously to work even if no clock is provided to the timer.
  - If the timer clock is still present, the dead-time generator is reactivated to drive the outputs with the level programmed in the CHxOIS and CHxNOIS bits after a dead-time. Even in this case, tmr_ocx and tmr_ocxn cannot be driven to their active level together.

    NOTE: Because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 CLK_TMR clock cycles).

  - If OSSI = 0, the timer releases the output control (taken over by the GPIO controller, which forces a Hi-Z state). Otherwise, the enable outputs remain or become high once one of the CHxEN or CHxNEN bits is high.
- The break status flag (SYSBRKIF, BRKIF and BRK2IF bits in the Timer INTSTS register) is set. An interrupt is generated if the BRKIE bit in the Timer DMAINTEN register is set. A DMA request can be sent if the UPDDE bit in the Timer DMAINTEN register is set.
- If the AOE bit in the Timer BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it 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 are active on the level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BRKIF and BRK2IF cannot be cleared.

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, tmr_ocx/tmr_ocxn polarities and state when disabled, CHxOCMODE configurations, break enable, and polarity). The application can choose from 3 levels of protection selected by the lock bits in the Timer BDTR register. Refer to break and dead-time register (BDTR).

The lock bits can be written only once after an MCU reset.

Figure 17-38 shows an example of the behavior of the outputs in response to a break.

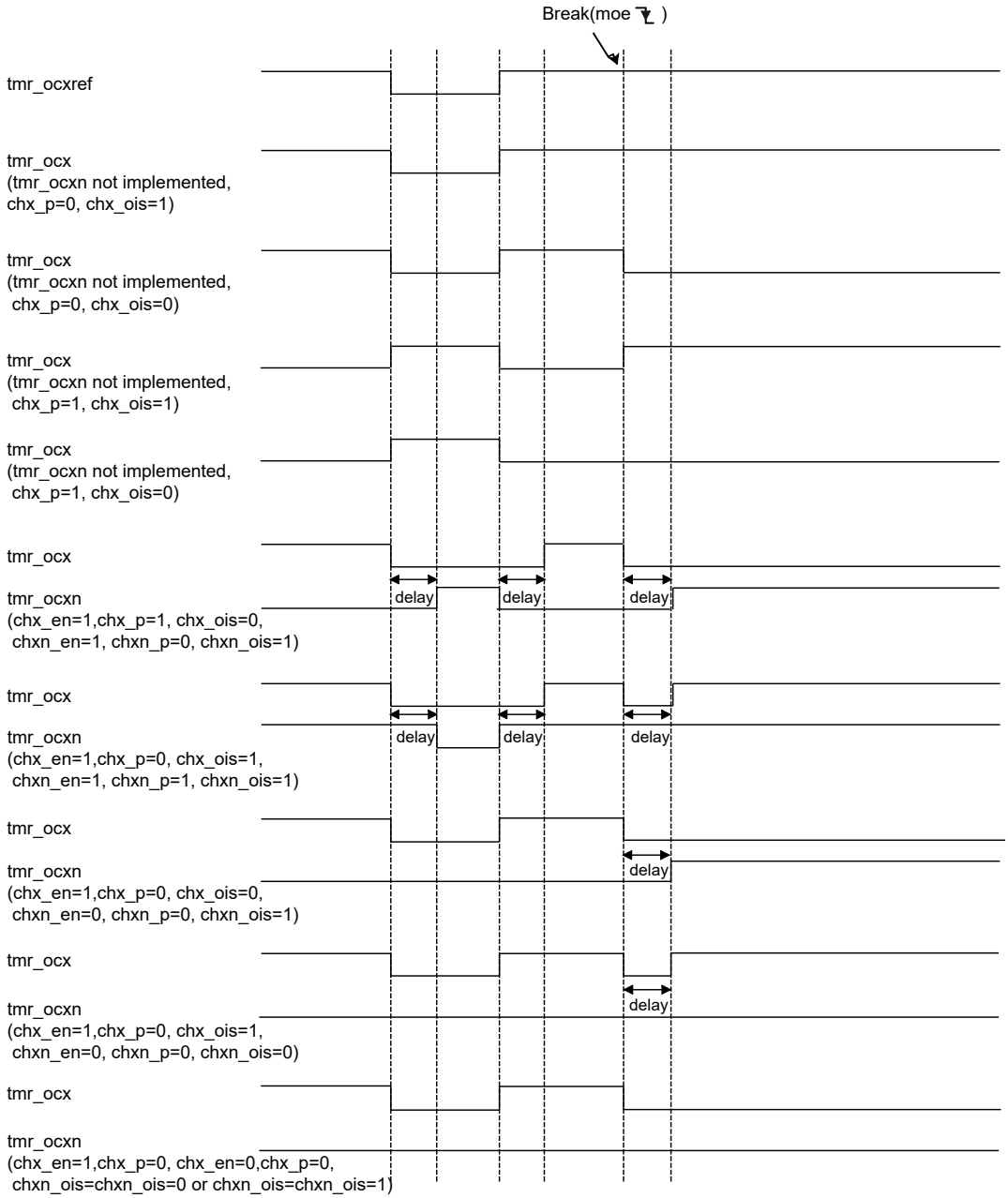


Figure 17-38 Various Output Behavior in Response to a Break Event on tmr_brk (OSSI = 1)

The two break inputs have different behaviors on timer outputs:

- The tmr_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- The tmr_brk2 can only disable (inactive state) the PWM outputs.

The tmr_brk has a higher priority than tmr_brk2 input, as described in Table 17-13.

NOTE: tmr_brk2 must only be used with OSSR = OSSI = 1.

Table 17-13 Behavior of Timer Outputs versus tmr_brk/tmr_brk2 Inputs

Tmr_brk	Tmr_brk2	Timer Outputs State	Typical Use Case	
			Tmr_ocxn Output (Low Side Switches)	Tmr_ocx Output (High Side Switches)
Active	X	• Inactive then forced output state (after a dead-time) • Outputs disabled if OSSI = 0 (Control taken over by GPIO logic)	ON after dead-time insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 17-39 gives an example of tmr_ocx and tmr_ocxn output behavior in case of active signals on tmr_brk and tmr_brk2 inputs. In this case, both outputs have active high polarities (CHxP = CHxNP = 0 in Timer CCER register).

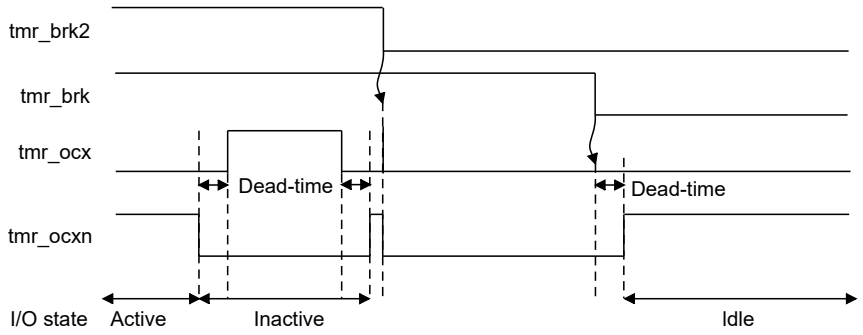


Figure 17-39 PWM output state Following tmr_brk and tmr_brk2 Assertion (OSSI = 1)

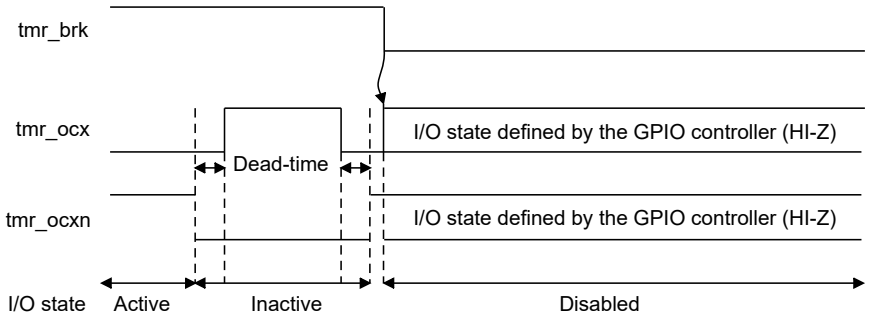


Figure 17-40 PWM Output State Following tmr_brk Assertion (OSSI = 0)

17.3.19 Clearing the Tmr_ocxref Signal on an External Event

The tmr_ocxref signal of a given channel can be cleared when a high level is applied on the tmr_ocref_clr_int input (CHxOCREFCLREN enable bit in the corresponding Timer CCMRx register set to 1). Tmr_ocxref remains low until the next update event (UEV) occurs. This function can only be used in output compare and PWM modes. It does not work in forced mode. tmr_ocref_clr_int input can be selected between the tmr_ocref_clr input and tmr_etrg_f (tmr_etrg after the filter) by configuring the OCREFCLRSEL bit in the INTSTS register.

The tmr_ocref_clr input can be selected among several inputs using the OCREFCLRSRCSEL[1:0] bit field in the Timer AF2 register, as shown on Figure 17-41. Refer to ADVTMR0/ADVTMR1/ ADVTMR2 Pins and Internal Signals for a list of sources available in the product.

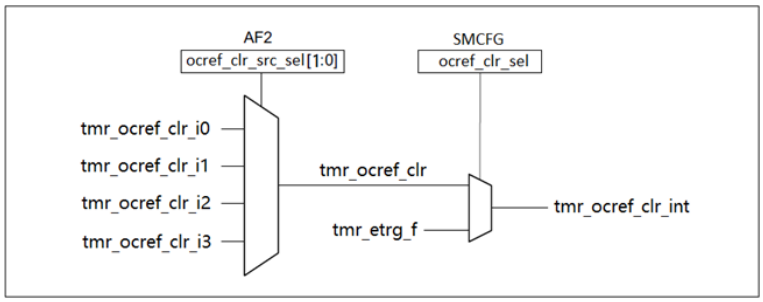


Figure 17-41 Tmr_ocref_clr Input Selection Multiplexer

When tmr_etrg_f is chosen, tmr_etrg must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETRGFDIVCFG[1:0] of the Timer SMCFG register set to '00'.
2. The external clock mode 1 must be disabled: bit EXTCLKMODE1EN of the Timer SMCFG register set to '0'.
3. The external trigger polarity (ETRGP) and the external trigger filter configuration (ETRGFILTCFG) can be configured according to application needs (as per polarity of the source connected to the trigger and the eventual need to remove noise using the filter).

Figure 17-42 shows the behavior of the tmr_ocxref signal when the tmr_etrg_f input becomes high for both values of the enable bit CHxOCREFCLREN. In this example, the ADVTMR is programmed in PWM mode.

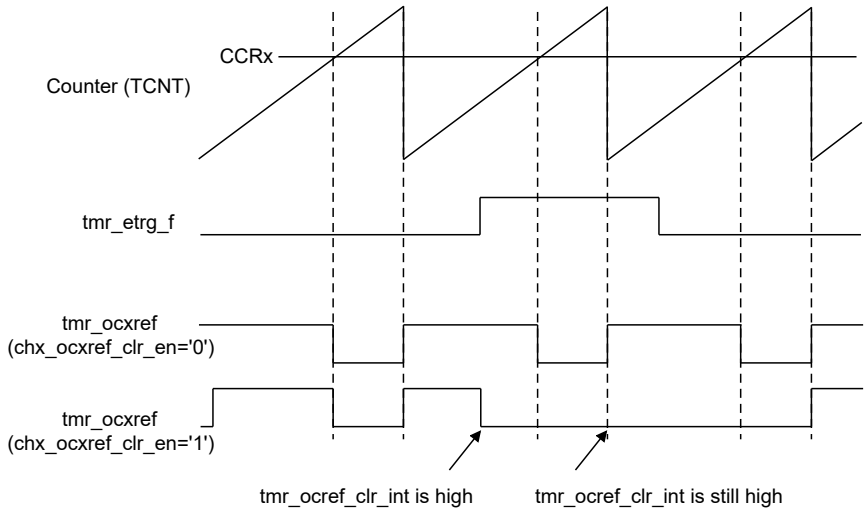


Figure 17-42 Clearing ADVTMR Tmr_ocxref

NOTE: If the duty cycle of a PWM signal is set to 100% (when the value of CCRx is greater than ARR), the tmr_ocxref is re-enabled when the counter overflows.

17.3.20 6-Step PWM Generation

When the complementary outputs are used on a channel, preload bits are available on the CHxOCMODE, CHxEN and CHxNEN bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can 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 SWCOMGEN bit in the SWEVTGEN register or by hardware (on tmr_itrg rising edge).

A flag is set when the COM event occurs (COMIF bit in the INTSTS register), which can generate an interrupt (if the COMIE bit is set in the Timer DMAINTEN register) or a DMA request (if the COMDE bit is set in the Timer DMAINTEN register).

Figure 17-43 describes the behavior of the tmr_ocx and tmr_ocxn outputs when a COM event occurs, in one examples of programmed configurations.

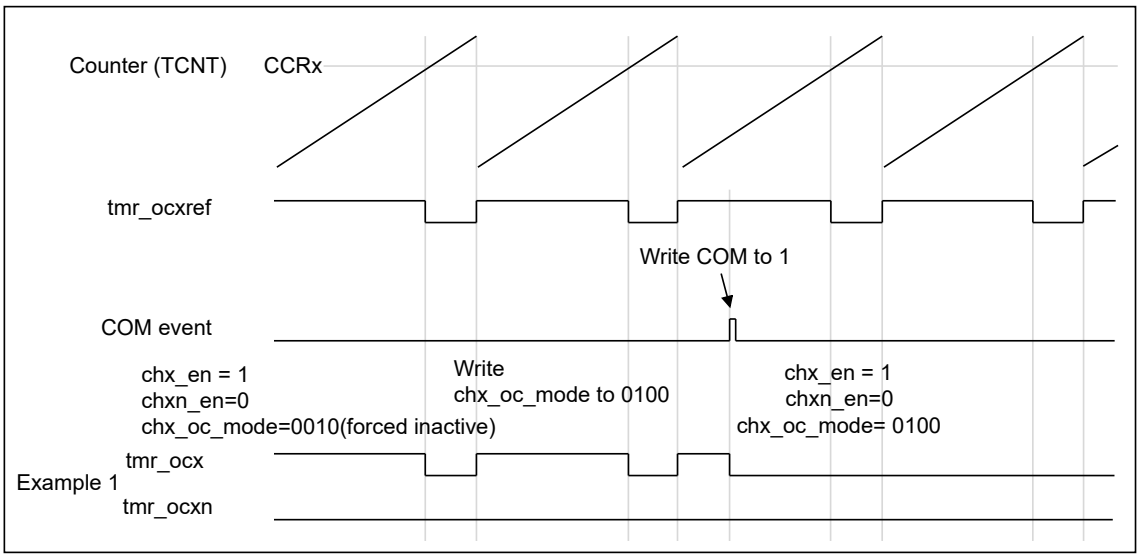


Figure 17-43 6-Step Generation, COM Example (OSSR = 1)

17.3.21 Single-Pulse Mode

Single-pulse mode 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. Single-pulse mode is selected by setting the SINGLEPLSMODE bit in the Timer CTRL1 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 up-counting: TCNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In down-counting: TCNT > CCRx

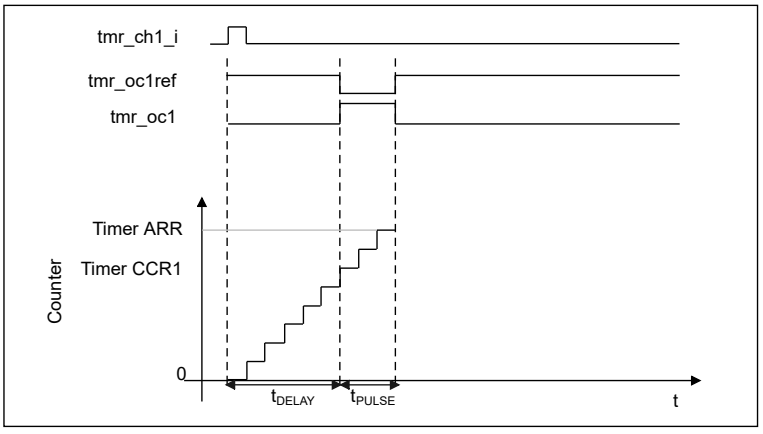


Figure 17-44 Single-Pulse Mode Example

For example, one may want to generate a positive pulse on tmr_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tmr_ch1 input pin.

When using tmr_ch1_fp as trigger 1:

- Map tmr_ch1_fp to tmr_ch1 by writing CH1MODESEL = '01' in the Timer CHXMODECTRL1 register.
- tmr_ch1_fp must detect a rising edge, write CH1P = '0' and CH1NP= '0' in the Timer CCER register.
- Configure tmr_ch1_fp as trigger for the slave mode controller (tmr_itrg) by writing ITRGSEL = 00110 in the Timer SMCFG register.
- tmr_ch1_fp is used to start the counter by writing SLVMODECTRL to '0110' in the Timer SMCFG register (trigger mode).

The single-pulse mode 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 Timer CCR0 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (ARR - CCR0).
- When building 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 PWM mode 2 must be enabled by writing CH0OCMODE = 111 in the Timer CHXMODECTRL1 register. Optionally the preload registers can be enabled by writing CH0CCRSHDWEN = '1' in the Timer CHXMODECTRL1 register and ARRSHDWEN in the Timer CTRL1 register. In this case one has to write the compare value in the Timer CCR0 register, the auto-reload value in the Timer ARR register, generate an update by setting the SWUPDGEN bit and wait for the external trigger event on tmr_ch1. CH1P is written to '0' in this example.

In the example, the DIR and TCNTALIGNMODE bits in the Timer CTRL1 register should be low. Since only single-pulse mode is needed, a 1 must be written in the SINGLEPLSMODE bit in the Timer CTRL1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When SINGLEPLSMODE bit in the Timer CTRL1 register is set to '0', the repetitive mode is selected.

In the case of tmr_ocx fast enable, when using single-pulse mode, the TCNTEN bit is set by the edge detection on the tmr_chx input, which enables the counter. The output then toggles based on the comparison between the counter and the compare value. However, there is a limitation on the minimum delay, t_{DELAY} min, due to the need for several clock cycles for these operations.

If the goal is to generate a waveform with the minimum delay, the CHxOCFASTEN bit can be enabled in the Timer CCMRx register. This will cause tmr_ocxref (and tmr_ocx) to be triggered immediately in response to the stimulus, without considering the comparison. The new level of tmr_ocxref will be the same as if a comparison match had occurred. It's important to note that CHxOCFASTEN only has an effect when the channel is configured in PWM1 or PWM2 mode.

17.3.22 Retriggerable Single-Pulse Mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable single-pulse mode described in Single-Pulse Mode:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in slave mode, with the bits SLVMODECTRL[3:0] = '1000' (combined reset + trigger mode) in the Timer SMCFG register, and the CHxOCMODE[3:0] bits set to '1000' or '1001' for retriggerable SINGLEPLSMODE mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

NOTE: For compatibility reasons, the CHxOCMODE[3:0] and SLVMODECTRL[3:0] bit fields are divided into two sections, with the most significant bit not being adjacent to the three least significant bits.

This mode must not be used with center-aligned PWM modes. It is mandatory to have TCNTALIGNMODE[1:0] = 00 in Timer CTRL1.

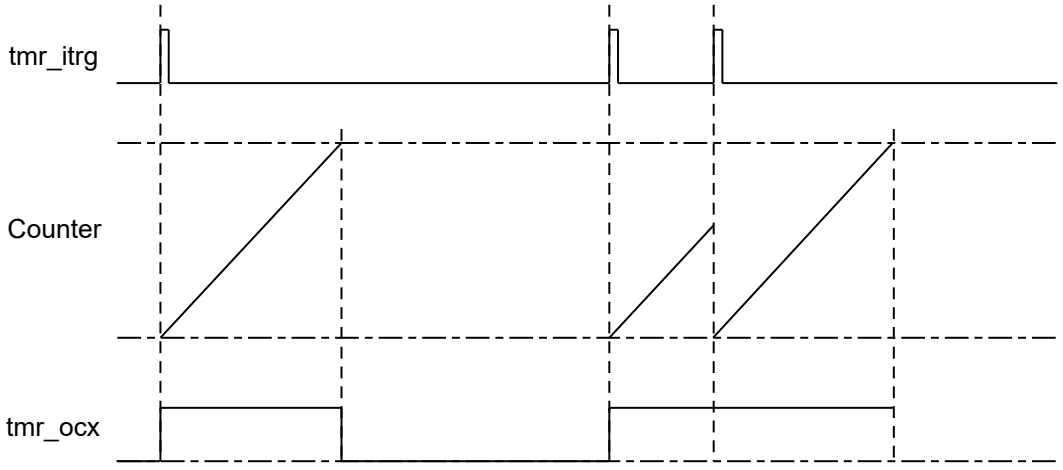


Figure 17-45 Retriggerable Single-Pulse Mode

17.3.23 Encoder Interface Mode

17.3.23.1 Quadrature Encoder

To select the encoder interface mode, you need to set the SLVMODECTRL value in the SMCFG register. Here are the options:

- 0001: Used to set when the counter counts on tmr_ch0 edges only.
- 0010: Used to set when the counter counts on tmr_ch1 edges only.
- 0011: Used to set when the counter counts on both tmr_ch0 and tmr_ch1 edges.

To configure the polarity of tmr_ch0 and tmr_ch1, you can set the CH0P and CH1P bits in the Timer CCER register. Additionally, if required, you can program the input filter. It is important to ensure that CH0NP and CH1NP are kept low.

The two inputs, tmr_ch0 and tmr_ch1, are used to interface to a quadrature encoder. Refer to Table 17-14. The counter is clocked by each valid transition on tmr_ch0fp or tmr_ch1fp (tmr_ch0 and tmr_ch1 after input filter and polarity selection, tmr_ch0fp = tmr_ch0 if not filtered and not inverted, tmr_ch1fp = tmr_ch1 if not filtered and not inverted) assuming that it is enabled (TCNTEN bit in Timer CTRL1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses and the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the Timer CTRL1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tmr_ch0 or tmr_ch1), whatever the counter is counting on tmr_ch0 only, tmr_ch1 only, or both tmr_ch0 and tmr_ch1.

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 Timer ARR register

(0 to ARR or ARR down to 0, depending on the direction). Therefore, the Timer ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, and 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 quadrature encoder and its content. Therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. Table 17-14 summarizes the possible combinations, assuming tmr_ch0 and tmr_ch1 do not switch simultaneously.

Table 17-14 Counting Direction versus Encoder Signals (CH0P = CH1P = 0)

Active Edge	SLVMODE CTRL[3:0]	Level on Opposite Signal (tmr_ch0fp0 for tmr_ch1, tmr_ch1fp1 for tmr_ch0)	tmr_ch0fp0 Signal		tmr_ch1fp1 signal	
			Rising	Falling	Rising	Falling
Counting on tmr_ch0 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tmr_ch1 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tmr_ch0 and tmr_ch1 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature 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 indicates the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

17.3.23.2 Encoder Clock Output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tmr_otrg output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode can be enabled by setting the MSTMODECTRL[3:0] bit field to 1000 in the Timer CTRL2 register. It is applicable for the following SLVMODECTRL[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SLVMODECTRL[3:0] code is prohibited and could result in unexpected behavior.

17.3.24 Direction Bit Output

It is possible to output a direction signal out of the timer on the output signals (copy of the DIR bit in the Timer CTRL1 register). This is achieved by setting the CHxOCMODE[3:0] bit field to 1011 in the Timer CHXMODECTRLx register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

17.3.25 UPDIF Bit Remapping

The UPDIFREMAP bit in the Timer CTRL1 register ensures that the update interrupt flag (UPDIF) is continuously copied into bit TCNT[31] of the timer counter register. This enables the atomic reading of both the counter value and the potential roll-over condition signaled by the UPDIFCPY flag. This feature simplifies calculations by preventing race conditions that may arise from shared processing between a background task (counter reading) and an interrupt (update interrupt). There is no latency between the UPDIF and UPDIFCPY flags assertion.

17.3.26 Timer Input XOR Function

The CH0SEL bit in the Timer CTRL2 register allows the input filter of channel 0 to be connected to the output of an XOR gate, combining the three input pins tmr_ch0, tmr_ch1 and tmr_ch2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per Figure 17-46.

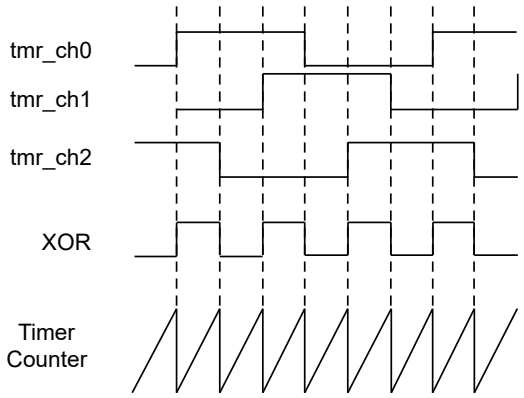


Figure 17-46 Measuring Time Interval between Edges on 3 Signals

17.3.27 Interfacing with Hall Sensors

The motor is driven by generating PWM signals using Advanced-Control Timers. In Figure 17-47, there is an additional timer called the interfacing timer. This timer captures the input from three timer input pins (tmr_ch0, tmr_ch1, and tmr_ch2) which are connected through an XOR to the tmr_ch0 input channel. The selection of the tmr_ch0 input channel is done by setting the CH0SEL bit in the Timer CTRL2 register.

The slave mode controller is configured in reset mode; the slave input is tmr_ch0fp_edg. 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 0 is configured in capture mode, and the capture signal is tmr_trc (see Figure 17-26). 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 that changes the configuration of the channels of the Advanced-Control Timer (by triggering a COM event). The Advanced-Control 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 through the tmr_otrg output.

For instance, if there is a change in the hall inputs connected to one of the timers, it is desired to modify the PWM configuration of the Advanced-Control Timer after a specified delay.

- Configure 3 timer inputs ORed to the tmr_ch0 input channel by writing the CH0SEL bit in the Timer CTRL2 register to '1'.
- Program the time base: Write the Timer ARR to the max value (the counter must be cleared by the tmr_ch0 change). Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors.
- Program the channel 0 in capture mode (tmr_trc selected): Write the CH0MODESEL bits in the Timer CHXMODECTRL1 register to '01'. The digital filter can also be programmed if needed.
- Program channel 2 in PWM 2 mode with the desired delay: Write the CH1OCMODE bits to '111' and the CH1MODESEL bits to '00' in the Timer CHXMODECTRL1 register.
- Select tmr_oc1ref as trigger output on tmr_otrg: Write the MSTMODECTRL1 bits in the Timer CTRL2 register to '101'.

In the Advanced-Control Timer, the right tmr_itrg input must be selected as a trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCSHWDEN = 1 in the Timer CTRL2 register), and the COM event is controlled by the trigger input (CCUPDSRC = 1 in the Timer CTRL2 register). The PWM control bits (CHxEN, CHxOCMODE) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tmr_oc1ref).

Figure 17-47 illustrates the hall sensor interface.

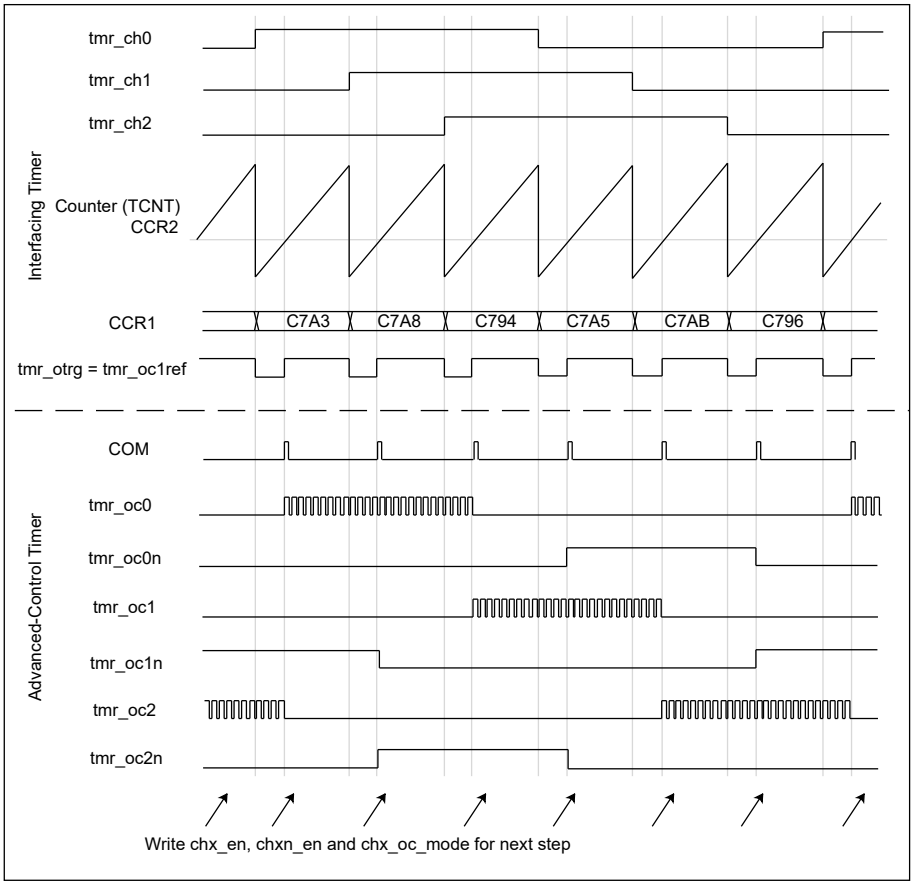


Figure 17-47 Example of Hall Sensor Interface

17.3.28 Timer Synchronization

The 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. They can be synchronized in several modes: reset mode, gated mode, trigger mode, reset + trigger and gated + reset modes. Figure 17-48 shows examples of master/slave timer connections.

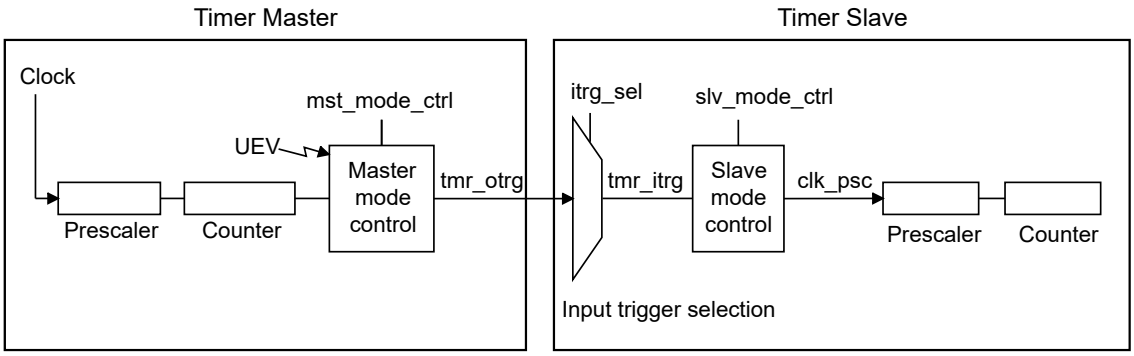


Figure 17-48 Timer Master and Slave Example

17.3.28.1 Slave Mode: Reset Mode

The counter and its prescaler can be reset when a trigger input event occurs. Additionally, if the UPDREQSRC bit in the Timer CTRL1 register is set to low, an update event (UEV) is triggered. This will update all the preloaded registers, including Timer ARR and Timer CCRx.

In the following example, the up-counter is reset when a rising edge is detected on the tmr_ch0 input:

- Configure channel 0 to detect rising edges on tmr_ch0. Configure the input filter duration (in this example, keep CH0ICFLT = 0000 since no filter is needed). The capture prescaler is not used for triggering, so it does not need to be configured. The CH0MODESEL bits select the input capture source only, CH0MODESEL = 01 in the Timer CHXMODECTRL1 register. Write CH0P = 0 and CH0NP = 0 in the Timer CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SLVMODECTRL = 0100 in the SMCFG register. Select tmr_ch0 as the input source by writing ITRGSEL = 00101 in the SMCFG register.
- Start the counter by writing TCNTEN = 1 in the Timer CTRL1 register.

The counter starts counting on the internal clock, then behaves normally until tmr_ch0 rising edge. When tmr_ch0 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TRGIF bit in the INTSTS register), and an interrupt request or a DMA request can be sent if enabled (depending on the TRGIE and TRGDE bits in the DMAINTEN register).

Figure 17-49 shows this behavior when the auto-reload register Timer ARR = 0x36. The delay between the rising edge on tmr_ch0 and the actual reset of the counter is due to the resynchronization circuit on tmr_ch0 input.

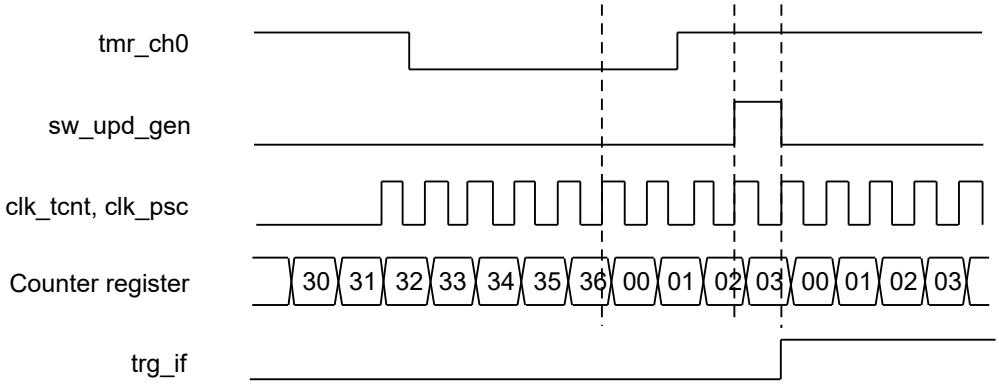


Figure 17-49 Control Circuit in Reset Mode

17.3.28.2 Slave Mode: Gated Mode

The counter can be enabled depending on the level of a selected input.

In the following example, the up-counter counts only when tmr_ch0 input is low:

- Configure channel 0 to detect low levels on tmr_ch0. Configure the input filter duration (in this example, we do not need any filter, so we keep CH0ICFLT = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CH0MODESEL bits select the input capture source only, CH0MODESEL = 01 in Timer CHXMODECTRL1 register. Write CH0P = 1 and CH0NP = 0 in Timer CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SLVMODECTRL = 0101 in SMCFG register. Select tmr_ch0 as the input source by writing ITRGSEL = 00101 in SMCFG register.
- Enable the counter by writing TCNTEN = 1 in the Timer CTRL1 register (in gated mode, the counter doesn't start if TCNTEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tmr_ch0 is low and stops as soon as tmr_ch0 becomes high. The TRGIF flag in the INTSTS register is set both when the counter starts or stops.

The delay between the rising edge on tmr_ch0 and the actual stop of the counter is due to the resynchronization circuit on tmr_ch0 input.

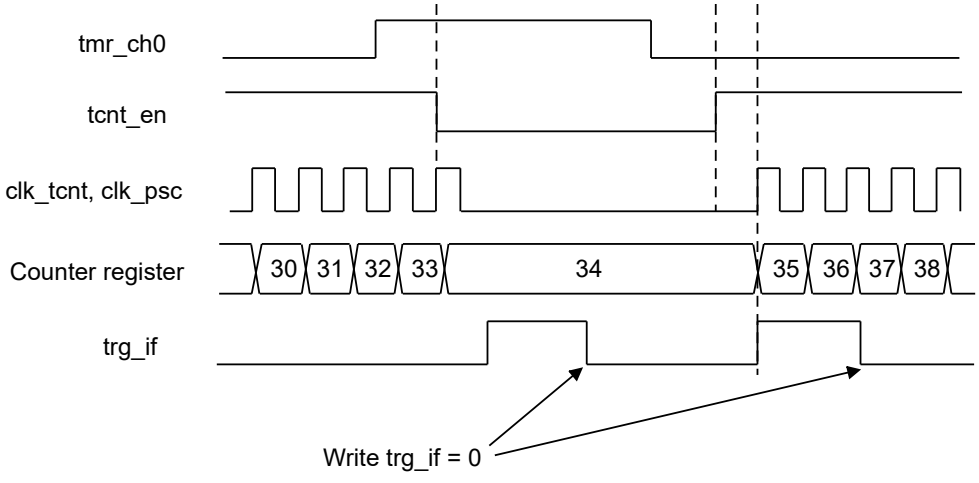


Figure 17-50 Control Circuit in Gated Mode

17.3.28.3 Slave Mode: Trigger Mode

The counter can start in response to an event on a selected input.

In the following example, the up-counter starts in response to a rising edge on tmr_ch1 input:

- Configure channel 1 to detect rising edges on tmr_ch1. Configure the input filter duration (in this example, keep CH1ICFLT = 0000 as no filter is needed). The capture prescaler is not used for triggering, so it does not need to be configured. The CH1MODESEL bits are configured to select the input capture source only, CH1MODESEL = 01 in Timer CHXMODECTRL1 register. Write CH1P = 1 and CH1NP = 0 in Timer CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SLVMODECTRL = 0110 in the SMCFG register. Select tmr_ch1 as the input source by writing ITRGSEL = 00110 in the SMCFG register.

When a rising edge occurs on tmr_ch1, the counter starts counting on the internal clock and the TRGIF flag is set.

The delay between the rising edge on tmr_ch1 and the actual start of the counter is due to the resynchronization circuit on tmr_ch1 input.

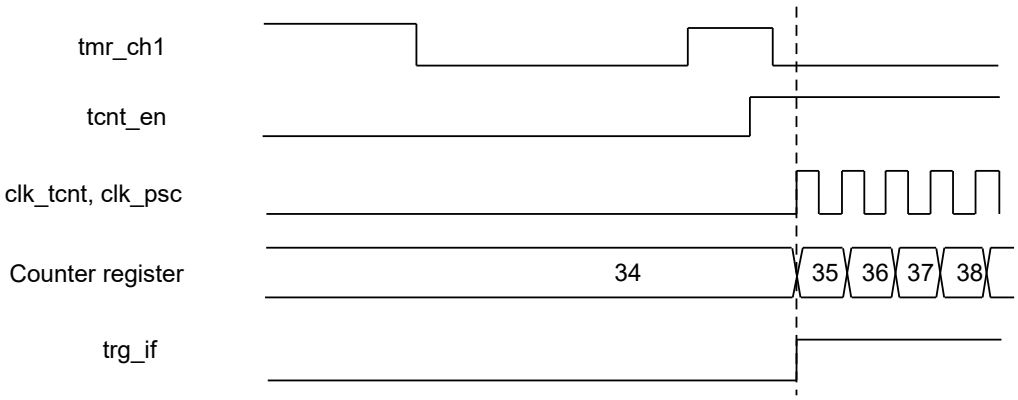


Figure 17-51 Control Circuit in Trigger Mode

17.3.28.4 Slave Mode: Combined Reset + Trigger Mode

In this scenario, when the selected trigger input (tmr_itrg) experiences a rising edge, the counter is reset, the registers are updated, and the counter begins. This mode is used for single-pulse mode.

17.3.28.5 Slave Mode: Combined Gated + Reset Mode

The counter clock is enabled when the trigger input (tmr_itrg) is high. The counter stops and is reset when the trigger goes low. The start and stop of the counter are controlled. This mode allows to detect out-of-range PWM signals, specifically when the duty cycle exceeds the maximum expected value.

17.3.28.6 Slave Mode: External Clock Mode 2 + Trigger Mode

You can use external clock mode 2 alongside another slave mode (excluding external clock mode 1 and encoder mode). In this configuration, the tmr_etrg_f signal serves as the external clock input, while another input can be chosen as the trigger input (in reset mode, gated mode, or trigger mode). It is advisable to avoid selecting tmr_etrg as tmr_etrg through the ITRGSEL bits of the SMCFG register.

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 tmr_etrg_f signal is used as the 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 tmr_etrg as tmr_etrg through the ITRGSEL bits of the SMCFG register.

In the following example, the up-counter is incremented at each rising edge of the tmr_etrg_f signal as soon as a rising edge of tmr_ch0 occurs:

1. Configure the external trigger input circuit by programming the SMCFG register as follows:
   - ETRGFILTCFG = 0000: No filter
   - ETRGFDIVCFG = 00: Prescaler disabled
   - ETRGP = 0: Detection of rising edges on tmr_etrg_f and EXTCLKMODE1EN = 1 to enable the external clock mode 1.
2. Configure channel 0 to detect rising edges on TI:
   - CH0ICFLT = 0000: No filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CH0MODESEL = 01 in the Timer CHXMODECTRL1 register to select only the input capture source
   - CH0P = 0 and CH0NP = 0 in the Timer CHXMODECTRL1 register to validate the polarity (and detect rising edge only).
3. Configure the tmr_in trigger mode by writing SLVMODECTRL = 0110 in the SMCFG register. Select tmr_ch0 as the input source by writing ITRGSEL = 00101 in the SMCFG register.

A rising edge on tmr_ch0 enables the counter and sets the TRGIF flag. The counter then counts on tmr_etrg rising edges.

The delay between the rising edge of the tmr_etrg_f signal and the actual reset of the counter is due to the resynchronization circuit on the tmr_etrg input.

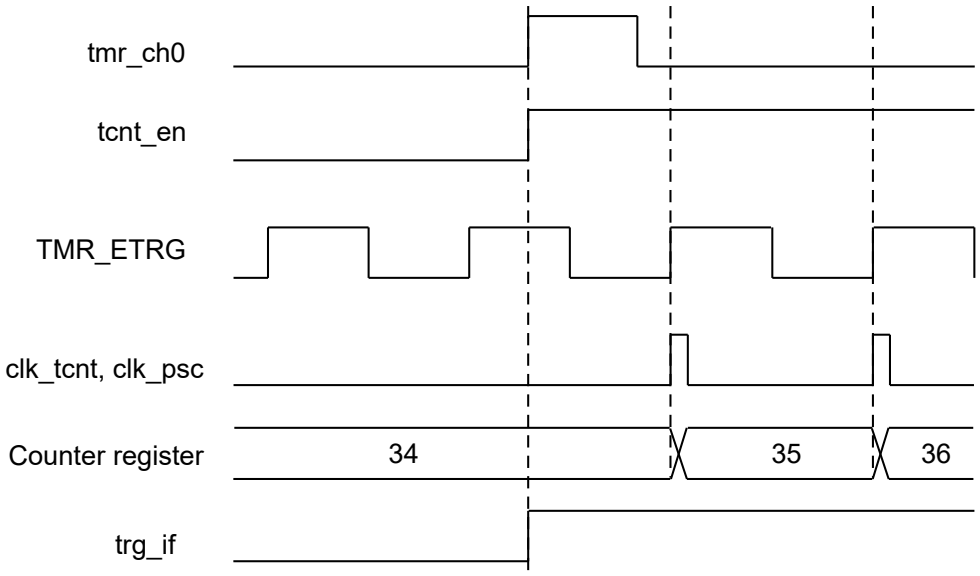


Figure 17-52 Control Circuit in External Clock Mode 2 + Trigger Mode

NOTE: The clock of the slave peripherals (timer, ADC, ...) receiving the tmr_otrg or the tmr_otrg2 signals must be enabled before receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

17.3.29 ADC Synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of ch3_ocref
- Rising edge on ch4_ocref or falling edge on ch5_ocref

The triggers are issued on the tmr_otrg2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MSTMODECTRL[3:0] bits in the Timer CTRL2 register.

An example of an application for 3-phase motor drives is given in Figure 17-34.

NOTE: The clock of the slave peripherals (timer, ADC, ...) receiving the tmr_otrg or the tmr_otrg2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

The clock of the ADC 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 timer.

17.3.30 DMA Burst Mode

The timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register Timer DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the Timer DMAR register is actually redirected to one of the timer registers.

The DMA burst length is determined by the DBL[5:0] bits within the Timer DMACFG register. To obtain more information about these bits, please refer to the description of the DBA bit in the Timer DMACFG register. When a read or write operation occurs at the Timer DMAR address, the timer recognizes it as a burst transfer and counts the number of transfers in half-words or bytes.

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 0, 1, 2, 3) upon an update event, 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.
   - The number of data to transfer = 3 (see note below).
   - Circular mode disabled.
2. Configure the DMACFG register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the Timer update DMA request (set the UPDDE bit in the Timer DMAINTEN register).
4. Enable Timer.
5. Enable the DMA channel.

This example demonstrates a scenario where each CCRx register needs to be updated once. However, if every CCRx register is to be updated twice, the total number of data to transfer should

be 6. For instance, consider a buffer in the RAM that contains data1, data2, data3, data4, data5, and data6. The data is then transferred to the CCRx registers in the following manner:

During the first update DMA request, data1 is transferred to CCR1, data2 is transferred to CCR2, and data3 is transferred to CCR3. On the second update DMA request, data4 is transferred to CCR1, data5 is transferred to CCR2, and data6 is transferred to CCR3.

NOTE: A null value can be written to the reserved registers.

17.3.31 DMA Requests

The ADVTMR0/ADVTMR1/ADVTMR2 can generate a DMA request, as shown in Table 17-15.

Table 17-15 DMA Request

DMA Request Signal	DMA Acronym	DMA Request	Enable Control Bit
tmr_upd_dma	TMR_UP	Update	UPDDE
tmr_ch0_dma	TMR_CH0	Capture/compare 0	CH0DE
tmr_ch1_dma	TMR_CH1	Capture/compare 1	CH1DE
tmr_ch2_dma	TMR_CH2	Capture/compare 2	CH2DE
tmr_ch3_dma	TMR_CH3	Capture/compare 3	CH3DE
tmr_com_dma	TMR_COM	Commutation (COM)	COMDE
tmr_trg_dma	TMR_ETRG	Trigger	TRGDE

17.4 Interrupts

The ADVTMR0/ADVTMR1/ADVTMR2 can generate multiple interrupts, as shown in Table 17-16.

Table 17-16 Interrupt Requests

Interrupt Acronym	Interrupt Event	Event Flag	Enable Control Bit	Interrupt Clear Method	Exit from Sleep Mode	Exit from Stop and Standby Mode
TMR_UPD	Update	UPDIF	UPDIE	Write 0 in UPDIF	Yes	No
TMR_CH0~CH3	Capture/compare 0	CH0IF	CH0IE	Write 0 in CH0IF	Yes	No
	Capture/compare 1	CH1IF	CH1IE	Write 0 in CH1IF	Yes	No
	Capture/compare 2	CH2IF	CH2IE	Write 0 in CH2IF	Yes	No
	Capture/compare 3	CH3IF	CH3IE	Write 0 in CH3IF	Yes	No
TMR_TRG_COM	Commutation (COM)	COMIF	COMIE	Write 0 in COMIF	Yes	No
	Trigger	TRGIF	TRGIE	Write 0 in TRGIF	Yes	No
TMR_DIR	Direction	DIRF	DIRIE	Write 0 in DIRF	Yes	No
TMR_BRK	Break	BRKIF	BRKIE	Write 0 in BRKIF	Yes	No
	Break2	BRK2IF		Write 0 in BRK2IF	Yes	No
	System Break	SYSBRKIF		Write 0 in SYSBRKIF	Yes	No

17.5 Registers

17.5.1 Register Address Map

Offset	Register Name	Register Description
0x0000	ADVTMR_CTRL1	TMR control register 1
0x0004	ADVTMR_CTRL2	TMR control register 2
0x0008	ADVTMR_SMCFG	TMR slave mode control register
0x000c	ADVTMR_DMAINTEN	TMR DMA/interrupt enable register
0x0010	ADVTMR_INTSTS	TMR status register
0x0014	ADVTMR_SWEVTGEN	TMR event generation register
0x0018	ADVTMR_CHXMODECTRL1	TMR capture/compare mode register 1
0x001c	ADVTMR_CHXMODECTRL2	TMR capture/compare mode register 2
0x0020	ADVTMR_CCER	TMR capture/compare enable register
0x0024	ADVTMR_TCNT	TMR counter
0x0028	ADVTMR_PSC	TMR prescaler
0x002c	ADVTMR_ARR	TMR auto-reload register
0x0030	ADVTMR_RCR	TMR repetition counter register
0x0034	ADVTMR_CCR0	TMR capture/compare register 0
0x0038	ADVTMR_CCR1	TMR capture/compare register 1
0x003c	ADVTMR_CCR2	TMR capture/compare register 2
0x0040	ADVTMR_CCR3	TMR capture/compare register 3
0x0044	ADVTMR_BDTR	TMR break and dead-time register

Offset	Register Name	Register Description
0x0048	ADVTMR_CCR4	TMR capture/compare register 4
0x004c	ADVTMR_CCR5	TMR capture/compare register 5
0x0050	ADVTMR_CHXMODECTRL3	TMR capture/compare mode register 3
0x0054	ADVTMR_CHXSEL	TMR timer input selection register
0x0058	ADVTMR_AF1	TMR alternate function option register 1
0x005c	ADVTMR_AF2	TMR alternate function option register 2
0x0060	ADVTMR_DMACFG	TMR DMA control register
0x0064	ADVTMR_DMAR	TMR DMA address for full transfer

17.5.2 Register Field Details

17.5.2.1 ADVTMR_CTRL1

0x0000			TMR control register 1											ADVTMR_CTRL1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				UPDIF REMAP	Reserv ed	CLKDTFDIV		ARRSH DWEN	TCNTALIGNMODE		DIR	SINGLE PLSMO DE	UPDRE QSRC	UPDDI S	TCNTE N
Type	RO				RW	RO	RW		RW	RW		RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-17 TMR Control Register 1 Description

Field	Name	Description
31:12	Reserved	Reserved
11	UPDIFREMAP	UPDIF status bit remapping 0: No remapping. UPDIF status bit is not copied to TCNT register bit 31. 1: Remapping enabled. UPDIF status bit is copied to TCNT register bit 31.
10	Reserved	Reserved
9:8	CLKDTFDIV	Clock division This bitfield indicates the division ratio between the timer clock (CLK_TMR) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (etrg, chx) 00: $t_{DTS} = t_{CLK\_TMR}$

Field	Name	Description
		01: t_{DTS} = 2 x t_{CLK_TMR} 10: t_{DTS}= 4 x t_{CLK_TMR} 11: Reserved, do not program this value
7	ARRSHDWEN	Auto-reload register shadow enable 0: TMR_ARR register is not buffered 1: TMR_ARR register is buffered
6:5	TCNTALIGNMODE	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 (CHxMODESEL = 00 in CHxMODECTRLx 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 (CHxMODESEL = 00 in CHxMODECTRLx 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 (CHxMODESEL = 00 in CHxMODECTRLx 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 (TCNTEN = 1).
4	DIR	Direction 0: Counter used as up-counter 1: Counter used as down-counter **NOTE:** This bit is read only when the timer is configured in center-aligned mode or encoder mode.
3	SINGLEPLSMODE	One pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit TCNTEN)

Field	Name	Description
2	UPDREQSRC	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 'SWUPDGEN' bit • Update generation through the slave mode controller 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.
1	UPDDIS	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 'SWUPDGEN' 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 'SWUPDGEN' bit is set or if a hardware reset is received from the slave mode controller.
0	TCNTEN	Counter enable 0: Counter disabled 1: Counter enabled **NOTE:** External clock, gated mode and encoder mode can work only if the 'TCNTEN' bit has been previously set by software.

Field	Name	Description
		However, trigger mode can set the 'TCNTEN' bit automatically by hardware. If SINGLEPLSMODE = 1, TCNTEN will be clear by hardware when UEV happens.

17.5.2.2 ADVTMR_CTRL2

0x0004			TMR control register 2											ADVTMR_CTRL2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved							MSTMODECTRL1BIT3	MSTMODECTRL2				Reserved	CH5OIS	Reserved	CH4OIS
Type	RO							RW	RW				RO	RW	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH3NOIS	CH3OIS	CH2NOIS	CH2OIS	CH1NOIS	CH1OIS	CH0NOIS	CH0OIS	CH0SEL	MSTMODECTRL1			CCDMAREQSRC	CCUPDSRC	Reserved	CCSHWDEN
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW			RW	RW	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-18 TMR Control Register 2 Description

Field	Name	Description
31:25	Reserved	Reserved
24	MSTMODECTRL1BIT3	The bit 3 of MSTMODECTRL1 Refer to [6:4]
23:20	MSTMODECTRL2	Master mode selection 2 These bits allow the information to be sent to ADC for synchronization (tmr_otrg2) to be selected. The combination is as follows: • 0000: Reset - the 'SWUPDGEN' bit from the SWEVTGEN register is used as trigger output (tmr_otrg2).

Field	Name	Description
		If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tmr_otrg2 is delayed compared to the actual reset. • 0001: Enable - the Counter Enable signal CNT_EN is used as trigger output (tmr_otrg2). It is useful to start several timers simultaneously or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the 'TCNTEN' 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 tmr_otrg2, except if the Master/Slave mode is selected (see the MSM bit description in ADVTMR_SMCFG register). • 0010: Update - the update event is selected as trigger output (tmr_otrg2). For instance, a master timer can then be used as a prescaler for a slave timer. • 0011: Compare pulse - the trigger output sends a positive pulse when the CH0IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tmr_otrg2). • 0100: Compare - tmr_oc0refc signal is used as trigger output (tmr_otrg2) • 0101: Compare - tmr_oc1refc signal is used as trigger output (tmr_otrg2) • 0110: Compare - tmr_oc2refc signal is used as trigger output (tmr_otrg2) • 0111: Compare - tmr_oc3refc signal is used as trigger output (tmr_otrg2) • 1000: Compare - tmr_oc4refc signal is used as trigger output (tmr_otrg2) • 1001: Compare - tmr_oc5refc signal is used as trigger output (tmr_otrg2) • 1010: Compare Pulse - tmr_oc3refc rising or falling edges generate pulses on tmr_otrg2 • 1011: Compare Pulse - tmr_oc5refc rising or falling edges generate pulses on tmr_otrg2

Field	Name	Description
		• 1100: Compare Pulse - tmr_oc3refc or tmr_oc5refc rising edges generate pulses on tmr_otrg2 • 1101: Compare Pulse - tmr_oc3refc rising or tmr_oc5refc falling edges generate pulses on tmr_otrg2 • 1110: Compare Pulse - tmr_oc4refc or tmr_oc5refc rising edges generate pulses on tmr_otrg2 • 1111: Compare Pulse - tmr_oc4refc rising or tmr_oc5refc falling edges generate pulses on tmr_otrg2 **NOTE:** The clock of the slave timer or ADC 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.
19	Reserved	Reserved
18	CH5OIS	CH5 output idle state Refer to CH0OIS.
17	Reserved	Reserved
16	CH4OIS	CH4 output idle state Refer to CH0OIS.
15	CH3NOIS	CH3N output idle state Refer to CH0NOIS.
14	CH3OIS	CH3 output idle state Refer to CH0OIS.
13	CH2NOIS	CH2N output idle state Refer to CH0NOIS.

Field	Name	Description
12	CH2OIS	CH2 output idle state Refer to CH0OIS.
11	CH1NOIS	CH1N output idle state Refer to CH0NOIS.
10	CH1OIS	CH1 output idle state Refer to CH0OIS.
9	CH0NOIS	CH0N output idle state 0: tmr_oc1n = 0 after a dead-time when MOE = 0 1: tmr_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 the BDTR register).
8	CH0OIS	CH0 output idle state 0: tmr_oc1 = 0 (after a dead-time) when MOE = 0 1: tmr_oc1=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 the BDTR register).
7	CH0SEL	TI0 selection 0: The tmr_ti0_in[7..0] multiplexer output is connected to tmr_ti0 input 1: tmr_ti0_in[7..0], tmr_ti1_in[7..0] and tmr_ti2_in[7..0] multiplexers outputs are XORed and connected to the tmr_ti0 input
6:4	MSTMODECTRL1	Master mode selection These bits select the information to be sent in master mode to slave timers for synchronization (tmr_otrg). The combination is as follows:

Field	Name	Description
		• 0000: Reset - the 'SWUPDGEN' bit from the SWEVTGEN register is used as trigger output (tmr_otrg). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tmr_otrg is delayed compared to the actual reset. • 0001: Enable - the Counter Enable signal CNT_EN is used as trigger output (tmr_otrg). It is useful to start several timers simultaneously or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between 'TCNTEN' 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 tmr_otrg, except if the master/slave mode is selected (see the MSM bit description in ADVTMR_SMCFG register). • 0010: Update - The update event is selected as trigger output (tmr_otrg). For instance, a master timer can then be used as a prescaler for a slave timer. • 0011: Compare Pulse - The trigger output sends a positive pulse when the CH0IF flag is to be set (even if it was already high) as soon as a capture or a compare match occurred (tmr_otrg). • 0100: Compare - tmr_oc0refc signal is used as trigger output (tmr_otrg) • 0101: Compare - tmr_oc1refc signal is used as trigger output (tmr_otrg) • 0110: Compare - tmr_oc2refc signal is used as trigger output (tmr_otrg) • 0111: Compare - tmr_oc3refc signal is used as trigger output (tmr_otrg) • 1000: Encoder Clock output - The encoder clock signal is used as trigger output (tmr_otrg). This code is valid for the following 'SLVMODECTRL'[3:0] values: 0001, 0010, 0011. Any other 'SLVMODECTRL'[3:0] code is not allowed and may lead to unexpected behavior. Other codes Reserved

Field	Name	Description
		NOTE: The clock of the slave timer or ADC 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.
3	CCDMAREQSRC	Capture/compare DMA selection 0: CCx DMA request sent when CCx event occurs 1: CCx DMA requests sent when an update event occurs
2	CCUPDSRC	Capture/compare control update selection 0: When capture/compare control bits are preloaded (CCSHWDEN = 1), they are updated by setting the SWCOMGEN bit only 1: When capture/compare control bits are preloaded (CCSHWDEN = 1), they are updated by setting the SWCOMGEN bit or when a rising edge occurs on tmr_itrg **NOTE:** This bit acts only on channels that have a complementary output.
1	Reserved	Reserved
0	CCSHWDEN	Capture/compare preloaded control 0: CHxEN, CHxNEN and CHxOCMODE bits are not preloaded 1: CHxEN, CHxNEN and CHxOCMODE bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (SWCOMGEN bit set or rising edge detected on tmr_itrg, depending on the CCUPDSRC bit). **NOTE:** This bit acts only on channels that have a complementary output.

17.5.2.3 ADVTMR_SMCFG

0x0008			TMR slave mode control register											ADVTMR_SMCFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										ITRGSELBIT4	ITRGSELBIT3	Reserved			OCREFCLRSEL
Type	RO										RW	RW	RO			RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ETRGP	EXTCLKMODE1EN	ETRGFDIVCFG		ETRGFILTCFG				MSM	ITRGSEL			SLVMODECTRL			
Type	RW	RW	RW		RW				RW	RW			RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-19 TMR Slave Mode Control Register Description

Field	Name	Description
31:22	Reserved	Reserved
21	ITRGSELBIT4	Trigger selection - bit 4 Refer to 'ITRGSEL'[2:0] description - bits 6:4
20	ITRGSELBIT3	Trigger selection - bit 3 Refer to 'ITRGSEL'[2:0] description - bits 6:4
19:17	Reserved	Reserved
16	OCREFCLRSEL	OCREF clear selection This bit is used to select the OCREF clear source. 0: tmr_ocref_clr_int is connected to the tmr_ocref_clr input

Field	Name	Description
		1: tmr_ocref_clr_int is connected to tmr_etrf
15	ETRGP	External trigger polarity This bit selects whether tmr_etr_in or tmr_etr_in invert is used for trigger operations 0: tmr_etr_in is non-inverted, active at high level or rising edge. 1: tmr_etr_in is inverted, active at low level or falling edge.
14	EXTCLKMODE1EN	External clock enable This bit enables external clock mode 1. 0: External clock mode 1 disabled 1: External clock mode 1 enabled. The counter is clocked by any active edge on the tmr_etrf signal. **NOTE:** Setting the 'EXTCLKMODE1EN' bit has the same effect as selecting external clock mode 0 with tmr_itrg connected to tmr_etrf (SLVMODECTRL = 0111 andITRGSEL = 00111). It is possible to simultaneously use external clock mode 1 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tmr_itrg must not be connected to tmr_etrf in this case ('ITRGSEL' bits must not be 00111). If external clock mode 0 and external clock mode 1 are enabled at the same time, the external clock input is tmr_etrf.
13:12	ETRGFDIVCFG	External trigger prescaler External trigger signal tmr_etrg_p frequency must be at most 1/4 of TMRCLK frequency. A prescaler can be enabled to reduce tmr_etrg_p frequency. It is useful when inputting fast external clocks on tmr_etr_in. 00: Prescaler OFF 01: tmr_etr_in frequency divided by 2 10: tmr_etr_in frequency divided by 4 11: tmr_etr_in frequency divided by 8

Field	Name	Description
11:8	ETRGFILTCFG	External trigger filter This bitfield then defines the frequency used to sample tmr_etrg_p signal and the length of the digital filter applied to tmr_etrg_p. 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: $f_{SAMPLING}$ = fclk_tmr, N = 2 0010: $f_{SAMPLING}$ = fclk_tmr, N = 4 0011: $f_{SAMPLING}$ = fclk_tmr, 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	MSM	Master/slave mode 0: No action 1: The effect of an event on the trigger input (tmr_itrg) is delayed to allow a perfect synchronization between the current timer and its slaves (through tmr_otrg). It is useful if we want to synchronize several timers on a single external event.
6:4	ITRGSEL	Trigger selection

Field	Name	Description
		This bitfield is combined with 'ITRGSEL'[4:3] bits. This bitfield selects the trigger input to be used to synchronize the counter. 00000: Internal trigger 0 (tmr_itr0) 00001: Internal trigger 1 (tmr_itr1) 00010: Internal trigger 2 (tmr_itr2) 00011: Internal trigger 3 (tmr_itr3) 00100: tmr_ti1 edge detector (tmr_ti1f_ed) 00101: Filtered timer input 1 (tmr_ch1fp0) 00110: Filtered timer input 2 (tmr_ti2fp2) 00111: External trigger input (tmr_etrf) 01000: Internal trigger 0 (tmr_itr4) 01001: Internal trigger 1 (tmr_itr5) 01010: Internal trigger 1 (tmr_itr6) 01011: Internal trigger 1 (tmr_itr7) 01100: Internal trigger 1 (tmr_itr8) 01101: Internal trigger 1 (tmr_itr9) 01110: Internal trigger 1 (tmr_itr10) 01111: Internal trigger 1 (tmr_itr11) 10000: Internal trigger 1 (tmr_itr12) 10001: Internal trigger 1 (tmr_itr13) 10010: Internal trigger 1 (tmr_itr14) 10011: Internal trigger 1 (tmr_itr15) Others: Reserved **NOTE:** These bits must be changed only when they are not used (for example, when SLVMODECTRL = 0000) to avoid wrong edge detections at the transition.
3:0	SLVMODECTRL	Slave mode selection

Field	Name	Description
		When external signals are selected the active edge of the trigger signal (tmr_itrg) is linked to the polarity selected on the external input (see input control register and control register description). 0000: Slave mode disabled - if TCNTEN = 1 then the prescaler is clocked directly by the internal clock. 0001: Quadrature encoder mode 1, x2 mode - Counter counts up/down on tmr_ch0fp0 edge depending on tmr_ti1fp2 level. 0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tmr_ti1fp2 edge depending on tmr_ch0fp0 level. 0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tmr_ch0fp0 and tmr_ti1fp2 edges depending on the level of the other input. 0100: Reset Mode - Rising edge of the selected trigger input (tmr_itrg) reinitializes the counter and generates an update of the registers. 0101: Gated mode - The counter clock is enabled when the trigger input (tmr_itrg) 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. 0110: Trigger mode - The counter starts at a rising edge of the trigger tmr_itrg (but it is not reset). Only the start of the counter is controlled. 0111: External clock mode 0 - Rising edges of the selected trigger (tmr_itrg) clock the counter. 1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tmr_itrg) reinitializes the counter, generates an update of the registers and starts the counter. 1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tmr_itrg) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Field	Name	Description
		NOTE: The gated mode must not be used if ch0_fp_ed is selected as the trigger input (ITRGSEL = 00100). Indeed, ch0_fp_ed outputs 1 pulse for each transition on TI0F, whereas the gated mode checks the level of the trigger signal. The clock of the slave peripherals (timer, ADC, ...) receiving the tmr_otrg or the tmr_otrg2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

17.5.2.4 ADVTMR_DMAINTEN

0x000c			TMR DMA/interrupt enable register											ADVTMR_DMAINTEN		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										DIRIE	Reserved				
Type	RO										RW	RO				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	TRGDE	COMDE	CH3DE	CH2DE	CH1DE	CH0DE	UPDDE	BRKIE	TRGIE	COMIE	CH3IE	CH2IE	CH1IE	CH0IE	UPDIE
Type	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-20 TMR DMA/Interrupt Enable Register Description

Field	Name	Description
31:22	Reserved	Reserved
21	DIRIE	Direction change interrupt enable 0: Direction Change interrupt disabled 1: Direction Change interrupt enabled
20:15	Reserved	Reserved
14	TRGDE	Trigger DMA request enable 0: Trigger DMA request disabled 1: Trigger DMA request enabled
13	COMDE	COM DMA request enable 0: COM DMA request disabled 1: COM DMA request enabled

Field	Name	Description
12	CH3DE	Capture/compare 3 DMA request enable 0: CC3 DMA request disabled 1: CC3 DMA request enabled
11	CH2DE	Capture/compare 2 DMA request enable 0: CC2 DMA request disabled 1: CC2 DMA request enabled
10	CH1DE	Capture/compare 1 DMA request enable 0: CC1 DMA request disabled 1: CC1 DMA request enabled
9	CH0DE	Capture/compare 0 DMA request enable 0: CC0 DMA request disabled 1: CC0 DMA request enabled
8	UPDDE	Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled
7	BRKIE	Break interrupt enable 0: Break interrupt disabled 1: Break interrupt enabled
6	TRGIE	Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
5	COMIE	COM interrupt enable 0: COM interrupt disabled 1: COM interrupt enabled
4	CH3IE	Capture/compare 3 interrupt enable

Field	Name	Description
		0: CC3 interrupt disabled 1: CC3 interrupt enabled
3	CH2IE	Capture/compare 2 interrupt enable 0: CC2 interrupt disabled 1: CC2 interrupt enabled
2	CH1IE	Capture/compare 1 interrupt enable 0: CC1 interrupt disabled 1: CC1 interrupt enabled
1	CH0IE	Capture/compare 0 interrupt enable 0: CC0 interrupt disabled 1: CC0 interrupt enabled
0	UPDIE	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

17.5.2.5 ADVTMR_INTSTS

0x0010			TMR status register											ADVTMR_INTSTS		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										DIRF	Reserved			CH5IF	CH4IF
Type	RO										RC_W0	RO			RC_W0	RC_W0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		SYSBRKIF	CH3OF	CH2OF	CH1OF	CH0OF	BRK2IF	BRKIF	TRGIF	COMIF	CH3IF	CH2IF	CH1IF	CH0IF	UPDIF
Type	RO		RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-21 TMR Status Register Description

Field	Name	Description
31:22	Reserved	Reserved
21	DIRF	Direction change interrupt flag This flag is set by hardware when the direction changes in encoder mode. It is cleared by software by writing it to '0'. 0: No direction change 1: Direction change
20:18	Reserved	Reserved
17	CH5IF	Compare 5 interrupt flag Refer to CH0IF description. **NOTE:** Channel 5 can only be configured as output.

Field	Name	Description
16	CH4IF	Compare 4 interrupt flag Refer to CH0IF description. **NOTE:** Channel 4 can only be configured as output.
15:14	Reserved	Reserved
13	SYSBRKIF	System break interrupt flag This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active. This flag must be reset to restart PWM operation. 0: No break event occurred. 1: An active level has been detected on the system break input. An interrupt is generated if BRKIE = 1 in the Timer DMAINTEN register.
12	CH3OF	Capture/compare 3 overcapture flag Refer to CH0OF description.
11	CH2OF	Capture/compare 2 overcapture flag Refer to CH0OF description.
10	CH1OF	Capture/compare 1 overcapture flag Refer to CH0OF description.
9	CH0OF	Capture/compare 0 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 CCR0 register while CH0IF flag was already set.
8	BRK2IF	Break 2 interrupt flag

Field	Name	Description
		This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is inactive. 0: No break event occurred. 1: An active level has been detected on the break 2 input. An interrupt is generated if BRKIE = 1 in the Timer DMAINTEN register.
7	BRKIF	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. An interrupt is generated if BRKIE = 1 in the Timer DMAINTEN register.
6	TRGIF	Trigger interrupt flag This flag is set by hardware on the TRG trigger event (active edge detected on tmr_itrg input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software. 0: No trigger event occurred. 1: Trigger interrupt pending.
5	COMIF	COM interrupt flag This flag is set by hardware on COM event (when capture/compare control bits - CHxEN, CHxNEN, CHxOCMODE - have been updated). It is cleared by software. 0: No COM event occurred 1: COM interrupt pending
4	CH3IF	Capture/compare 3 interrupt flag Refer to CH0IF description.
3	CH2IF	Capture/compare 2 interrupt flag

Field	Name	Description
		Refer to CH0IF description.
2	CH1IF	Capture/compare 1 interrupt flag Refer to CH0IF description.
1	CH0IF	Capture/compare 0 interrupt flag This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the CCR0 register (input capture mode only). 0: No compare match / no input capture occurred 1: A compare match or an input capture occurred If channel CH0 is configured as output: this flag is set when the content of the counter TCNT matches the content of the CCR0 register. When the content of CCR0 is greater than ARR's, the CH0IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode. Refer to the 'TCNTALIGNMODE' bits in the CTRL1 register for the full description. If channel CH0 is configured as input: this bit is set when the counter value has been captured in the CCR0 register (an edge has been detected on IC0, as per the edge sensitivity defined with the 'CH0P' and 'CH0NP' bits setting, in CCER).
0	UPDIF	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 UPDDIS = 0 in the CTRL1 register. • When CNT is reinitialized by software using the 'SWUPDGEN' bit in the SWEVTGEN register, if UPDREQSRC = 0 and UPDDIS = 0 in the CTRL1 register.

Field	Name	Description
		• When CNT is reinitialized by a trigger event if UPDREQSRC = 0 and UPDDIS = 0 in the CTRL1 register.

17.5.2.6 ADVTMR_SWEVTGEN

0x0014			TMR event generation register											ADVTMR_SWEVTGEN		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							SWBRK 2GEN	SWBRK GEN	SWTRG GEN	SWCO MGEN	SWCH3 CCGEN	SWCH2 CCGEN	SWCH1 CCGEN	SWCH0 CCGEN	SWUPD GEN
Type	RO							WC	WC	WC	WC	WC	WC	WC	WC	WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-22 TMR Event Generation Register Description

Field	Name	Description
31:9	Reserved	Reserved
8	SWBRK2GEN	Break 2 generation This bit is set by software to generate an event; it is automatically cleared by hardware. 0: No action 1: A break 2 event is generated. MOE bit is cleared and BRK2IF flag is set. Related interrupt can occur if enabled.
7	SWBRKGEN	Break generation This bit is set by software to generate an event; it is automatically cleared by hardware. 0: No action 1: A break event is generated. MOE bit is cleared and BRKIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Field	Name	Description
6	SWTRGGEN	Trigger generation This bit is set by software to generate an event, it is automatically cleared by hardware. 0: No action 1: The TRGIF flag is set in INTSTS register. Related interrupt or DMA transfer can occur if enabled.
5	SWCOMGEN	Capture/compare control update generation This bit can be set by software; it is automatically cleared by the hardware. 0: No action 1: When 'CCSHWDEN' bit is set, it allows the update of CHxEN, CHxNEN and CHxOCMODE bits. **NOTE:** This bit acts only on channels having a complementary output.
4	SWCH3CCGEN	Capture/compare 3 generation Refer to SWCH0CCGEN description
3	SWCH2CCGEN	Capture/compare 2 generation Refer to SWCH0CCGEN description
2	SWCH1CCGEN	Capture/compare 1 generation Refer to SWCH0CCGEN description
1	SWCH0CCGEN	Capture/compare 0 generation This bit is set by software 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: the CH0IF flag is set, corresponding interrupt or DMA request is sent if enabled. • If channel CC1 is configured as input:

Field	Name	Description
		the current value of the counter is captured in the CCR0 register. The CH0IF flag is set, and the corresponding interrupt or DMA request is sent if enabled. The CH0OF flag is set if the CH0IF flag is already high.
0	SWUPDGEN	Update generation This bit can be set by software; it is automatically cleared by hardware. 0: No action 1: Reinitialize the counter and generate 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 (up-counting), else it takes the auto-reload value (ARR) if DIR = 1 (down-counting).

17.5.2.7 ADVTMR_CHXMODECTRL1

0x0018			TMR capture/compare mode register 1											ADVTMR_CHXMODECTRL1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved							CCMR1								
Type	RO							RW								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CCMR1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-23 TMR capture/compare Mode Register 1 Description

Field	Name	Description
31:25	Reserved	Reserved
24:0	CCMR1	The field of this register will have different functions when the channel is in different modes (input capture/output compare) **Output compare mode**: [24]: CH1OCMODE_bit3; the bit3 of CH1OCMODE [16]: CH0OCMODE_bit3; the bit3 of CH0OCMODE [15]: CH1OCREFCLREN; CH1 Output Compare OCxREF clear enable. Refer to CH0OCREFCLREN. [14:12]: CH1OCMODE; CH1 Output Compare mode selection. Refer to CH0OCMODE. [11]: CH1CCRSHDWEN; CH1 Output Compare shadow enable. Refer to CH0CCRSHDWEN. [10]: CH1OCFASTEN; CH1 Output Compare fast enable. Refer to CH0OCFASTEN. [9:8]: CH1MODESEL; CH1 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH1 channel is configured as output

Field	Name	Description
		01: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_ch1 10: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_ch0 11: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through the 'ITRGSEL' bit (SMCFG register). **NOTE:** CH1MODESEL bits are writable only when the channel is off (CH1EN = '0' in CCER). [7]: CH0OCREFCLREN; CH0 Output Compare OCxREF clear enable 0: tmr_oc1ref is not affected by the tmr_ocref_clr_int signal 1: tmr_oc1ref is cleared as soon as a High level is detected on tmr_ocref_clr_int signal (tmr_ocref_clr input or tmr_etrf input) [6:4]: CH0OCMODE; CH0 Output Compare mode selection These bits define the behavior of the output reference signal tmr_oc0ref from which tmr_oc0 and tmr_oc0n are derived. tmr_oc0ref is active high whereas tmr_oc0 and tmr_oc0n active level depends on 'CH0P' and 'CH0NP' bits. 0000: Frozen - The comparison between the output compare register CCR0 and the counter TCNT has no effect on the outputs. (This mode is used to generate a timing base). 0001: Set channel 0 to active level on match. tmr_oc0ref signal is forced high when the counter TCNT matches the capture/compare register 0 (CCR0). 0010: Set channel 0 to inactive level on match. tmr_oc0ref signal is forced low when the counter TCNT matches the capture/compare register 0 (CCR0). 0011: Toggle - tmr_oc0ref toggles when TCNT = CCR0. 0100: Force inactive level - tmr_oc0ref is forced low. 0101: Force active level - tmr_oc0ref is forced high. 0110: PWM mode 1 - In up-counting, channel 0 is active as long as TCNT is less than CCR0 else inactive. In down-counting, channel 0 is inactive (tmr_oc0ref = '0') as long as TCNT is great than CCR0 else active (tmr_oc0ref = '1'). 0111: PWM mode 2 - In up-counting, channel 0 is inactive as long as TCNT less than CCR0 else active. In down-counting, channel 0 is active as long as TCNT is great than CCR0 else inactive.

Field	Name	Description
		1000: Retrigerrable SINGLEPLSMODE mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become inactive at the next update. 1001: Retrigerrable SINGLEPLSMODE mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 2 and the channels become inactive at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become active again at the next update. 1010: Reserved 1011: Direction output. The tim_oc0ref signal is overridden by a copy of the DIR bit 1100: Combined PWM mode 1 - tmr_oc0ref has the same behavior as in PWM mode 1. tmr_oc0refc is the logical OR between tmr_oc0ref and tmr_oc1ref. 1101: Combined PWM mode 2 - tmr_oc0ref has the same behavior as in PWM mode 2. tmr_oc0refc is the logical AND between tmr_oc0ref and tmr_oc1ref. 1110: Asymmetric PWM mode 1 - tmr_oc0ref has the same behavior as in PWM mode 1. tmr_oc0refc outputs tmr_oc0ref when the counter is counting up, tmr_oc1ref when it is counting down. 1111: Asymmetric PWM mode 2 - tmr_oc0ref has the same behavior as in PWM mode 2. tmr_oc0refc outputs tmr_oc0ref when the counter is counting up, tmr_oc1ref when it is counting down. **NOTE:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in the BDTR register) and CH0MODESEL = '00' (the channel is configured in output). In PWM mode, 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. On channels having a complementary output, this bit field is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register then the 'CH0OCMODE' active bits take the new value from the preloaded bits only when a COM event is generated.

Field	Name	Description
		[3]: CH0CCRSHDWEN; CH0 Output Compare shadow enable 0: Preload register on CCR0 disabled. CCR0 can be written at any time, the new value is taken into account immediately. 1: Preload register on CCR0 enabled. Read/write operations access the preload register. CCR0 preload value is loaded in the active register at each update event. **NOTE:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in the BDTR register) and CH0MODESEL = '00' (the channel is configured in output). The PWM mode can be used without validating the preload register only in one pulse mode ('SINGLEPLSMODE' bit set in CTRL1 register). Else the behavior is not guaranteed. [2]: CH0OCFASTEN; CH0 Output Compare fast enable This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in single-pulse mode ('SINGLEPLSMODE' bit set in CTRL1 register), to have the output pulse starting as soon as possible after the starting trigger. 0: CC1 behaves normally depending on counter and CCR0 values even when the trigger is on. The minimum delay to activate CH0 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 CH0 output. Then, OC is set to the compare level independently from the result of the comparison. The delay to sample the trigger input and to activate CH0 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode. [1:0]: CH0MODESEL; CH0 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH0 channel is configured as output 01: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_ch0 10: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_ch1 11: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through 'ITRGSEL' bit (SMCFG register).

Field	Name	Description
		NOTE: CH0MODESEL bits are writable only when the channel is off (CH0EN = '0' in CCER).
		Input capture mode: [15:12]: CH1ICFLT; CH1 Input capture filter. Refer to CH0ICFLT [11:10]: CH1ICPSC; CH1 Input capture prescaler. Refer to CH0ICPSC [9:8]: CH1MODESEL; CH1 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH1 channel is configured as output 01: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_ch1 10: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_ch0 11: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through the 'ITRGSEL' bit (SMCFG register). **NOTE:** CH1MODESEL bits are writable only when the channel is off (CH1EN = '0' in CCER). [7:4]: CH0ICFLT; CH0 Input capture filter This bitfield defines the frequency used to sample tmr_ti1 input and the length of the digital filter applied to tmr_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: $f_{SAMPLING}$ = fclk_tmr, N = 2 0010: $f_{SAMPLING}$ = fclk_tmr, N = 4 0011: $f_{SAMPLING}$ = fclk_tmr, 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

Field	Name	Description
		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]: CH0ICPSC; CH0 Input capture prescaler This bitfield defines the ratio of the prescaler acting on CH0 input (tmr_ic0). The prescaler is reset as soon as CH0EN = '0' (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]: CH0MODESEL: CH0 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH0 channel is configured as output 01: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_ch0 10: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_ch1 11: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through 'ITRGSEL' bit (SMCFG register). **NOTE:** CH0MODESEL bits are writable only when the channel is off (CH0EN = '0' in CCER).

17.5.2.8 ADVTMR_CHXMODECTRL2

0x001c			TMR capture/compare mode register 2											ADVTMR_CHXMODECTRL2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved							CCMR2								
Type	RO							RW								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CCMR2															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-24 TMR capture/compare Mode Register 2 Description

Field	Name	Description
31:25	Reserved	Reserved
24:0	CCMR2	The field of this register will have different functions when the Channel is in different modes (input capture/output compare) **Output compare mode**: [24]: CH3OCMODE bit3; The bit3 of CH3OCMODE [16]: CH2OCMODE bit3; The bit3 of CH2OCMODE [15]: CH3OCREFCLREN; CH3 Output Compare OCxREF clear enable. Refer to CH2OCREFCLREN. [14:12]: CH3OCMODE; CH3 Output Compare mode selection. Refer to CH2OCMODE. [11]: CH3CCRSHDWEN; CH3 Output Compare shadow enable. Refer to CH2CCRSHDWEN. [10]: CH3OCFASTEN; CH3 Output Compare fast enable. Refer to CH2OCFASTEN. [9:8]: CH3MODESEL; CH3 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH3 channel is configured as output 01: CH3 channel is configured as input, tmr_ic3 is mapped on tmr_ch3

Field	Name	Description
		10: CH3 channel is configured as input, tmr_ic3 is mapped on tmr_ch2 11: CH3 channel is configured as input, tmr_ic3 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through the 'ITRGSEL' bit (SMCFG register) **NOTE:** CH3MODESEL bits are writable only when the channel is off (CH3EN = '0' in CCER). [7]: CH2OCREFCLREN; CH2 Output Compare OCxREF clear enable 0: tmr_oc2ref is not affected by the tmr_ocref_clr_int signal 1: tmr_oc2ref is cleared as soon as a High level is detected on tmr_ocref_clr_int signal (tmr_ocref_clr input or tmr_etrf input) [6:4]: CH2OCMODE; CH2 Output Compare mode selection These bits define the behavior of the output reference signal tmr_oc2ref from which tmr_oc2 and tmr_oc2n are derived. tmr_oc2ref is active high whereas tmr_oc2 and tmr_oc2n active level depends on 'CH2P' and 'CH2NP' bits. 0000: Frozen - The comparison between the output compare register CCR2 and the counter TCNT has no effect on the outputs. (This mode is used to generate a timing base). 0001: Set channel 2 to active level on match. tmr_oc2ref signal is forced high when the counter TCNT matches the capture/compare register 2 (CCR2). 0010: Set channel 2 to inactive level on match. tmr_oc2ref signal is forced low when the counter TCNT matches the capture/compare register 2 (CCR2). 0011: Toggle - tmr_oc2ref toggles when TCNT = CCR2. 0100: Force inactive level - tmr_oc2ref is forced low. 0101: Force active level - tmr_oc2ref is forced high. 0110: PWM mode 1 - In up-counting, channel 2 is active as long as TCNT is less than CCR2 else inactive. In down-counting, channel 2 is inactive (tmr_oc2ref = '0') as long as TCNT is great than CCR2 else active (tmr_oc2ref = '1'). 0111: PWM mode 2 - In up-counting, channel 2 is inactive as long as TCNT less than CCR2 else active. In down-counting, channel 2 is active as long as TCNT is great than CCR2 else inactive. 1000: Retrigerrable SINGLEPLSMODE mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become active again

Field	Name	Description
		at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become inactive at the next update. 1001: Retrigerrable SINGLEPLSMODE mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 2 and the channels become inactive at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become active again at the next update. 1010: Reserved 1011: Direction output. The tim_oc2ref signal is overridden by a copy of the DIR bit." 1100: Combined PWM mode 1 - tmr_oc2ref has the same behavior as in PWM mode 1. tmr_oc2refc is the logical OR between tmr_oc2ref and tmr_oc3ref. 1101: Combined PWM mode 2 - tmr_oc2ref has the same behavior as in PWM mode 2. tmr_oc2refc is the logical AND between tmr_oc2ref and tmr_oc3ref. 1110: Asymmetric PWM mode 1 - tmr_oc2ref has the same behavior as in PWM mode 1. tmr_oc2refc outputs tmr_oc2ref when the counter is counting up, tmr_oc3ref when it is counting down. 1111: Asymmetric PWM mode 2 - tmr_oc2ref has the same behavior as in PWM mode 2. tmr_oc2refc outputs tmr_oc2ref when the counter is counting up, tmr_oc3ref when it is counting down. **NOTE:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in the BDTR register) and CH2MODESEL = '00' (the channel is configured in output). In PWM mode, 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. On channels having a complementary output, this bit field is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register, then the 'CH2OCMODE' active bits take the new value from the preloaded bits only when a COM event is generated. [3]: CH2CCRSHDWEN; CH2 Output Compare shadow enable 0: Preload register on CCR2 disabled. CCR2 can be written at any time, the new value is taken into account immediately. 1: Preload register on CCR2 enabled. Read/write operations access the preload register. CCR2 preload value is loaded in the active register at each update event.

Field	Name	Description
		NOTE: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in the BDTR register) and CH2MODESEL = '00' (the channel is configured in output). The PWM mode can be used without validating the preload register only in one pulse mode ('SINGLEPLSMODE' bit set in CTRL1 register). Else the behavior is not guaranteed. [2]: CH2OCFASTEN; CH2 Output Compare fast enable This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in Single-pulse mode ('SINGLEPLSMODE' bit set in CTRL1 register), to have the output pulse starting as soon as possible after the starting trigger. 0: CC1 behaves normally depending on counter and CCR2 values even when the trigger is on. The minimum delay to activate CH2 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 CH2 output. Then, OC is set to the compare level independently from the result of the comparison. The delay to sample the trigger input and to activate CH2 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode. [1:0]: CH2MODESEL; CH2 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH2 channel is configured as output 01: CH2 channel is configured as input, tmr_ic2 is mapped on tmr_ch2 10: CH2 channel is configured as input, tmr_ic2 is mapped on tmr_ch3 11: CH2 channel is configured as input, tmr_ic2 is mapped on tmr_trc. This mode is working only if an internal trigger input is selected through 'ITRGSEL' bit (SMCFG register) **NOTE:** CH2MODESEL bits are writable only when the channel is off (CH2EN = '0' in CCER). **Input capture mode**: [15:12]: CH3ICFLT; CH3 Input capture filter. Refer to CH2ICFLT [11:10]: CH3ICPSC; CH3 Input capture prescaler. Refer to CH2ICPSC [9:8]: CH3MODESEL; CH3 I/O mode selection

Field	Name	Description
		This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH3 channel is configured as output 01: CH3 channel is configured as input, tmr_ic3 is mapped on tmr_ch3 10: CH3 channel is configured as input, tmr_ic3 is mapped on tmr_ch2 11: CH3 channel is configured as input, tmr_ic3 is mapped on tmr_trc. This mode is working only if an internal trigger input is selected through the 'ITRGSEL' bit (SMCFG register) **NOTE:** CH3MODESEL bits are writable only when the channel is off (CH3EN = '0' in CCER). [7:4]: CH2ICFLT; CH2 Input capture filter This bitfield defines the frequency used to sample tmr_ti1 input and the length of the digital filter applied to tmr_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: $f_{SAMPLING}$ = fclk_tmr, N = 2 0010: $f_{SAMPLING}$ = fclk_tmr, N = 4 0011: $f_{SAMPLING}$ = fclk_tmr, 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]: CH2ICPSC; CH2 Input capture prescaler

Field	Name	Description
		This bitfield defines the ratio of the prescaler acting on CH2 input (tmr_ic2). The prescaler is reset as soon as CH2EN = '0' (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]: CH2MODESEL:CH2 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH2 channel is configured as output 01: CH2 channel is configured as input, tmr_ic2 is mapped on tmr_ch2 10: CH2 channel is configured as input, tmr_ic2 is mapped on tmr_ch3 11: CH2 channel is configured as input, tmr_ic2 is mapped on tmr_trc. This mode is working only if an internal trigger input is selected through 'ITRGSEL' bit (SMCFG register). **NOTE:** CH2MODESEL bits are writable only when the channel is off (CH2EN = '0' in CCER).

17.5.2.9 ADVTMR_CCER

0x0020			TMR capture/compare enable register											ADVTMR_CCER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										CH5P	CH5EN	Reserved		CH4P	CH4EN
Type	RO										RW	RW	RO		RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH3NP	CH3NEN	CH3P	CH3EN	CH2NP	CH2NEN	CH2P	CH2EN	CH1NP	CH1NEN	CH1P	CH1EN	CH0NP	CH0NEN	CH0P	CH0EN
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-25 TMR capture/compare Enable Register Description

Field	Name	Description
31:22	Reserved	Reserved
21	CH5P	CH5 capture/compare polarity. Refer to CH0P description
20	CH5EN	CH5 capture/compare enable. Refer to CH0EN description
19:18	Reserved	Reserved
17	CH4P	CH4 capture/compare polarity. Refer to CH0P description
16	CH4EN	CH4 capture/compare enable. Refer to CH0EN description
15	CH3NP	CH3 capture/compare complementary output polarity. Refer to CH0NP description
14	CH3NEN	CH3 capture/compare complementary output enable. Refer to CH0NEN description
13	CH3P	CH3 capture/compare polarity. Refer to CH0P description

Field	Name	Description
12	CH3EN	CH3 capture/compare enable. Refer to CH0EN description
11	CH2NP	CH2 capture/compare complementary output polarity. Refer to CH0NP description
10	CH2NEN	CH2 capture/compare complementary output enable. Refer to CH0NEN description
9	CH2P	CH2 capture/compare polarity. Refer to CH0P description
8	CH2EN	CH2 capture/compare enable. Refer to CH0EN description
7	CH1NP	CH1 capture/compare complementary output polarity. Refer to CH0NP description
6	CH1NEN	CH1 capture/compare complementary output enable. Refer to CH0NEN description
5	CH1P	CH1 capture/compare polarity. Refer to CH0P description
4	CH1EN	CH1 capture/compare enable. Refer to CH0EN description
3	CH0NP	CH0 capture/compare complementary output polarity CH0 channel configured as output: 0: tmr_oc0n active high. 1: tmr_oc0n active low. CH0 channel configured as input: This bit is used in conjunction with CH0P to define the polarity of tmr_ch0fp0 and tmr_ch1fp0. Refer to CH0P description. **NOTE:** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in the BDTR register) and CH0MODESEL = '00' (channel configured as output). On channels having a complementary output, this bit is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register, then the CH0NP active bit takes the new value from the preloaded bit only when a commutation event is generated.
2	CH0NEN	CH0 capture/compare complementary output enable

Field	Name	Description
		0: Off - tmr_oc0n is not active. tmr_oc1n level is then function of MOE, OSSI, OSSR, OIS0, OIS0N and CH0EN bits. 1: On - tmr_oc0n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS0, OIS0N and CH0EN bits. **NOTE:** On channels having a complementary output, this bit is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register, then the CH0NEN active bit takes the new value from the preloaded bit only when a commutation event is generated.
1	CH0P	CH0 capture/compare polarity 0: OC0 active high (output mode) / Edge sensitivity selection (input mode, see below) 1: OC0 active low (output mode) / Edge sensitivity selection (input mode, see below) When CH0 channel is configured as input, both CH0NP/CH0P bits select the active polarity of CH0FP0 and CH1FP0 for trigger or capture operations. CH0NP = 0, CH0P = 0: Non-inverted/rising edge. The circuit is sensitive to CHxFP0 rising edge (capture or trigger operations in reset, external clock or trigger mode), CHxFP0 is not inverted (trigger operation in gated mode or encoder mode). CH0NP = 0, CH0P = 1: inverted/falling edge. The circuit is sensitive to CHxFP0 falling edge (capture or trigger operations in reset, external clock or trigger mode), CHxFP0 is inverted (trigger operation in gated or encoder mode). CH0NP = 1, CH0P = 1: non-inverted/both edges/ the circuit is sensitive to both CHxFP0 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), CHxFP0 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode. CH0NP = 1, CH0P = 0: the configuration is reserved, it must not be used. **NOTE:** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in the BDTR register).

Field	Name	Description
		On channels having a complementary output, this bit is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register, then the CH0P active bit takes the new value from the preloaded bit only when a commutation event is generated.
0	CH0EN	CH0 capture/compare enable 0: Capture mode disabled / OC0 is not active (see below) 1: Capture mode enabled / OC0 signal is output on the corresponding output pin When the CH0 channel is configured as output, the OC0 level depends on MOE, OSSI, OSSR, OIS0, OIS0N and CH0N_EN bits, regardless of the CH0EN bits state. **NOTE:** On channels having a complementary output, this bit is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register, then the CH0EN active bit takes the new value from the preloaded bit only when a commutation event is generated.

17.5.2.10 ADVTMR_TCNT

0x0024			TMR counter											ADVTMR_TCNT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	UPDIFCPY	Reserved														
Type	RO	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TCNT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-26 TMR Counter Description

Field	Name	Description
31	UPDIFCPY	UPDIF copy This bit is a read-only copy of the UPDIF bit of the INTSTS register. If the 'UPDIFREMAP' bit in the CTRL1 is reset, bit 31 is Reserved and read at 0.
30:16	Reserved	Reserved
15:0	TCNT	Counter value The register holds the counter value.

17.5.2.11 ADVTMR_PSC

0x0028			TMR prescaler											ADVTMR_PSC		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PSC															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-27 TMR Prescaler Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	PSC	Prescaler value The counter clock frequency (ftmr_cnt_ck) is equal to ftmr_psc_ck / (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 'SWUPDGEN' bit of SWEVTGEN register or through trigger controller when configured in reset mode).

17.5.2.12 ADVTMR_ARR

0x002c			TMR auto-reload register											ADVTMR_ARR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ARR															
Type	RW															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 17-28 TMR Auto-reload Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	ARR	Auto-reload value ARR is the value to be loaded in the actual auto-reload register.

17.5.2.13 ADVTMR_RCR

0x0030			TMR repetition counter register											ADVTMR_RCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	REPCNT															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	REPVAL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-29 TMR Repetition Counter Register Description

Field	Name	Description
31:16	REPCNT	Repetition counter real value
15:0	REPVAL	Repetition counter reload value This bitfield defines the update rate of the compare registers (that is, periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable. When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the 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

17.5.2.14 ADVTMR_CCR0

0x0034			TMR capture/compare register 0											ADVTMR_CCR0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH0CCRVAL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-30 TMR Capture/compare Register 0 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CH0CCRVAL	CH0 Capture/compare value If channel CH0 is configured as output: CCR0 is the value to be loaded in the actual capture/compare 0 register (preload value). It is loaded permanently if the preload feature is not selected in the CHxMODECTRL1 register (bit CH0CCRSHDWEN). Else the preload value is copied in the active capture/compare 0 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TCNT and signaled on tmr_oc0 output. If channel CH0 is configured as input: CCR0 is the counter value transferred by the last input capture 0 event (tmr_ic0). The CCR0 register is read-only and cannot be programmed.

17.5.2.15 ADVTMR_CCR1

0x0038			TMR capture/compare register 1											ADVTMR_CCR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH1CCRVAL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-31 TMR Capture/compare Register 1 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CH1CCRVAL	CH1 capture/compare value. Refer to CCR0.

17.5.2.16 ADVTMR_CCR2

0x003c			TMR capture/compare register 2											ADVTMR_CCR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH2CCRVAL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-32 TMR Capture/compare Register 2 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CH2CCRVAL	CH2 capture/compare value. Refer to CCR0.

17.5.2.17 ADVTMR_CCR3

0x0040	TMR capture/compare register 3													ADVTMR_CCR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH3CCRVAL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-33 TMR Capture/compare Register 3 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CH3CCRVAL	CH3 capture/compare value. Refer to CCR0.

17.5.2.18 ADVTMR_BDTR

0x0044			TMR break and dead-time register											ADVTMR_BDTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved						BRK2P	BRK2EN	BRK2FILTCFG				BRKFILTCFG			
Type	RO						RW	RW	RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MOE	AOE	BRKP	BRKEN	OSSR	OSSI	LOCK		DTG							
Type	RW	RW	RW	RW	RW	RW	RW		RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-34 TMR Break and Dead-Time Register Description

Field	Name	Description
31:26	Reserved	Reserved
25	BRK2P	Break 2 polarity 0: Break input tmr_brk2 is active low 1: Break input tmr_brk2 is active high **NOTE:** This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register). Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
24	BRK2EN	Break 2 enable This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources).

Field	Name	Description
		0: Break2 function disabled 1: Break2 function enabled **NOTE:** The BRKIN2 must only be used with OSSR = OSSI = 1. This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in the BDTR register). Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
23:20	BRK2FILTCFG	Break 2 filter This bitfield defines the frequency used to sample tmr_brk2 input and the length of the digital filter applied to tmr_brk2. 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, tmr_brk2 acts asynchronously 0001: $f_{SAMPLING} = f_{clk\_tmr}$, N = 2 0010: $f_{SAMPLING} = f_{clk\_tmr}$, N = 4 0011: $f_{SAMPLING} = f_{clk\_tmr}$, 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

Field	Name	Description
		NOTE: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in the BDTR register).
19:16	BRKFILTCFG	Break filter This bitfield defines the frequency used to sample tmr_brk input and the length of the digital filter applied to tmr_brk. 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, tmr_brk acts asynchronously 0001: $f_{SAMPLING} = f_{clk\_tmr}$, N = 2 0010: $f_{SAMPLING} = f_{clk\_tmr}$, N = 4 0011: $f_{SAMPLING} = f_{clk\_tmr}$, 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 **NOTE:** This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in the BDTR register).

Field	Name	Description
15	MOE	Main output enable This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tmr_brk or tmr_brk2). It is set by software or automatically depending on the AOE bit. It acts only on the channels which are configured in the output. 0: In response to a break 2 event. OC and OCN outputs are disabled in response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit. 1: OC and OCN outputs are enabled if their respective enable bits are set (CHxEN, CHxNEN in CCER register).
14	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 none of the break inputs tmr_brk and tmr_brk2 is active) **NOTE:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
13	BRKP	Break polarity 0: Break input tmr_brk is active low 1: Break input tmr_brk is active high **NOTE:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register). Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
12	BRKEN	Break enable This bit enables complete break protection (including all sources connected to bk_acth and BKIN sources).

Field	Name	Description
		0: Break function disabled 1: Break function enabled **NOTE:** This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in the BDTR register). Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
11	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. 0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state). 1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CHxEN = 1 or CHxNEN = 1 (the output is still controlled by the timer). **NOTE:** This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in the BDTR register).
10	OSSI	Off-state selection for idle mode This bit is used when MOE = 0 due to a break event or by a software write on channels configured as outputs. 0: When inactive, OC/OCN outputs are disabled (the timer releases the output control, which is taken over by the GPIO logic and imposes a Hi-Z state). 1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the dead-time. The timer maintains its control over the output. **NOTE:** This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in the BDTR register).

Field	Name	Description
9:8	LOCK	Lock configuration These bits provide write protection against software errors. 00: LOCK OFF - No bit is write protected. 01: LOCK Level 1 = DTG bits in BDTR register, OISx and OISxN bits in CTRL2 register and BRKEN/BRKP/AOE bits inBDTR register can no longer be written. 10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CHxP/CHxNP bits in CCER register, as long as the related channel is configured in output through the 'CHxMODESEL' bits) as well as OSSR and OSSI bits can no longer be written. 11: LOCK Level 3 = LOCK Level 2 + CC Control bits (CHxOCMODE and OCxPE bits in CHxMODECTRLx registers, as long as the related channel is configured in output through the 'CHxMODESEL' bits) can no longer be written. **NOTE:** The LOCK bits can be written only once after the reset. Once the BDTR register has been written, their content is frozen until the next reset.
7:0	DTG	Dead-time generator setup This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT corresponds 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 $T_{dtg} = 2xt_{DTS}$. DTG[7:5] = 110 — DT = (32+DTG[4:0])x t_{dtg} with $T_{dtg} = 8xt_{DTS}$. DTG[7:5] = 111 — DT = (32+DTG[4:0])x t_{dtg} with $T_{dtg} = 16xt_{DTS}$. Example if TDTS=125ns (8 MHz), dead-time possible values are: 0 to 15875ns by 125ns steps 16us to 31750ns by 250ns steps 32us to 63us by 1us steps, 64us to 126us by 2us steps

Field	Name	Description
		NOTE: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the BDTR register).

17.5.2.19 ADVTMR_CCR4

0x0048			TMR capture/compare register 4											ADVTMR_CCR4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	GC4C2	GC4C1	GC4C0	Reserved												
Type	RW	RW	RW	RO												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH4CCRVAL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-35 TMR Capture/compare Register 4 Description

Field	Name	Description
31	GC4C2	Group channel 4 and channel 2 distortion on channel 2 output: 0: No effect of tmr_oc4ref on tmr_oc2refc 1: tmr_oc2refc is the logical AND of tmr_oc2ref and tmr_oc4ref This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in CHxMODECTRL2). **NOTE:** It is also possible to apply this distortion on combined PWM signals.
30	GC4C1	Group channel 4 and channel 1 distortion on channel 1 output: 0: No effect of oc4ref on oc1refc 1: oc1refc is the logical AND of oc1ref and oc4ref This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in CHxMODECTRL1).

Field	Name	Description
		NOTE: It is also possible to apply this distortion on combined PWM signals.
29	GC4C0	Group channel 4 and channel 0 distortion on channel 0 output: 0: No effect of tmr_oc4ref on tmr_oc0refc 1: tmr_oc0refc is the logical AND of tmr_oc0ref and tmr_oc4ref This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in CHxMODECTRL1). **NOTE:** It is also possible to apply this distortion on combined PWM signals.
28:16	Reserved	Reserved
15:0	CH4CCRVAL	CH4 capture/compare value. Refer to CCR0.

17.5.2.20 ADVTMR_CCR5

0x004c			TMR capture/compare register 5											ADVTMR_CCR5		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH5CCRVAL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-36 TMR capture/compare Register 5 Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CH5CCRVAL	CH5 capture/compare value. Refer to CCR0

17.5.2.21 ADVTMR_CHXMODECTRL3

0x0050			TMR capture/compare mode register 3											ADVTMR_CHXMODECTRL3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved							CH5OCMODEBIT3	Reserved							CH4OCMODEBIT3
Type	RO							RW	RO							RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH5OCREFCLREN	CH5OCMODE			CH5CCRSHDWEN	CH5OCFASTEN	Reserved		CH4OCREFCLREN	CH4OCMODE			CH4CCRSHDWEN	CH4OCFASTEN	Reserved	
Type	RW	RW			RW	RW	RO		RW	RW			RW	RW	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-37 TMR Capture/compare Mode Register 3 Description

Field	Name	Description
31:25	Reserved	Reserved
24	CH5OCMODEBIT3	The bit3 of CH5OCMODE
23:17	Reserved	Reserved
16	CH4OCMODEBIT3	The bit3 of CH4OCMODE
15	CH5OCREFCLREN	CH5 Output Compare OCxREF clear enable. Refer to CH0OCREFCLREN
14:12	CH5OCMODE	CH5 Output Compare mode selection. Refer to CH0OCMODE
11	CH5CCRSHDWEN	CH5 Output Compare shadow enable. Refer to CH0CCRSHDWEN
10	CH5OCFASTEN	CH5 Output Compare fast enable. Refer to CH0OCFASTEN

Field	Name	Description
9:8	Reserved	Reserved
7	CH4OCREFCLREN	CH4 Output Compare OCxREF clear enable. Refer to CH0OCREFCLREN
6:4	CH4OCMODE	CH4 Output Compare mode selection. Refer to CH0OCMODE
3	CH4CCRSHDWEN	CH4 Output Compare shadow enable. Refer to CH0CCRSHDWEN
2	CH4OCFASTEN	CH4 Output Compare fast enable. Refer to CH0OCFASTEN
1:0	Reserved	Reserved

17.5.2.22 ADVTMR_CHXSEL

0x0054			TMR timer input selection register											ADVTMR_CHXSEL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved					CH3SRCSEL			Reserved					CH2SRCSEL		
Type	RO					RW			RO					RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					CH1SRCSEL			Reserved					CH0SRCSEL		
Type	RO					RW			RO					RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-38 TMR Timer Input Selection Register Description

Field	Name	Description
31:27	Reserved	Reserved
26:24	CH3SRCSEL	Selects tmr_ch3[0..7] input 000: tmr_ch3_in0: TMR_CH3(PAD) 0 01: tmr_ch3_in1 ... 111: tmr_ch3_in7
18:16	CH2SRCSEL	Selects tmr_ch2[0..7] input 000: tmr_ch2_in0: TMR_ch2(PAD) 001: tmr_ch2_in1 ... 111: tmr_ch2_in7
15:11	Reserved	Reserved

Field	Name	Description
10:8	CH1SRCSEL	Selects tmr_ch1[0..7] input 000: tmr_ch1_in0: TMR_ch1(PAD) 001: tmr_ch1_in1 ... 111: tmr_ch1_in7
7:3	Reserved	Reserved
2:0	CH0SRCSEL	Selects tmr_ch0[0..7] input 000: tmr_ch0_in0: tmr_ch0(PAD) 001: tmr_ch0_in1 ... 111: tmr_ch0_in7

17.5.2.23 ADVTMR_AF1

0x0058			TMR alternate function option register 1											ADVTMR_AF1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								SYSBRKEN				Reserved	ETRGSEL		
Type	RO								RW				RO	RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				BRKCOMP3P	BRKCOMP2P	BRKCOMP1P	BRKINP	Reserved				BRKCOMP3EN	BRKCOMP2EN	BRKCOMP1EN	BRKINEN
Type	RO				RW	RW	RW	RW	RO				RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 17-39 TMR Alternate Function Option Register 1 Description

Field	Name	Description
31:24	Reserved	Reserved
23:20	SYSBRKEN	System Break enable [0]: sbrkin[0] enable, high active [1]: sbrkin[1] enable, high active [2]: sbrkin[2] enable,high active [3]: sbrkin[3] enable, high active **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
19	Reserved	Reserved
18:16	ETRGSEL	etr_in source selection These bits select the etr_in input source. 000: tmr_etr0: TMR_ETRG input

Field	Name	Description
		001: tmr_etr1 ... 111: tmr_etr7 **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
15:12	Reserved	Reserved
11	BRKCOMP3P	tmr_brk_cmp3 input polarity This bit selects the tmr_brk_cmp3 input sensitivity. It must be programmed together with the BRKP polarity bit. 0: tmr_brk_cmp3 input polarity is not inverted (active low if BRKP = 0, active high if BRKP = 1) 1: tmr_brk_cmp3 input polarity is inverted (active high if BRKP = 0, active low if BRKP = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
10	BRKCOMP2P	tmr_brk_cmp2 input polarity This bit selects the tmr_brk_cmp2 input sensitivity. It must be programmed together with the BRKP polarity bit. 0: tmr_brk_cmp2 input polarity is not inverted (active low if BRKP = 0, active high if BRKP = 1) 1: tmr_brk_cmp2 input polarity is inverted (active high if BRKP = 0, active low if BRKP = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
9	BRKCOMP1P	tmr_brk_cmp1 input polarity This bit selects the tmr_brk_cmp1 input sensitivity. It must be programmed together with the BRKP polarity bit. 0: tmr_brk_cmp1 input polarity is not inverted (active low if BRKP = 0, active high if BRKP = 1) 1: tmr_brk_cmp1 input polarity is inverted (active high if BRKP = 0, active low if BRKP = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
8	BRKINP	TMR_BKIN input polarity This bit selects the TMR_BKIN alternate function input sensitivity. It must be programmed together with the BRKP polarity bit. 0: TMR_BKIN input polarity is not inverted (active low if BRKP = 0, active high if BRKP = 1)

Field	Name	Description
		1: TMR_BKIN input polarity is inverted (active high if BRKP = 0, active low if BRKP = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
7:4	Reserved	Reserved
3	BRKCOMP3EN	tmr_brk_cmp3 enable This bit enables the tmr_brk_cmp3 for the timer's tmr_brk input. tmr_brk_cmp3 output is 'ORed' with the other tmr_brk sources. 0: tmr_brk_cmp3 input disabled 1: tmr_brk_cmp3 input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
2	BRKCOMP2EN	tmr_brk_cmp2 enable
1	BRKCOMP1EN	tmr_brk_cmp1 enable This bit enables the tmr_brk_cmp1 for the timer's tmr_brk input. tmr_brk_cmp1 output is 'ORed' with the other tmr_brk sources. 0: tmr_brk_cmp1 input disabled 1: tmr_brk_cmp1 input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
0	BRKINEN	TMR_BKIN input enable This bit enables the TMR_BKIN alternate function input for the timer's tmr_brk input. TMR_BKIN input is 'ORed' with the other tmr_brk sources. 0: TMR_BKIN input disabled 1: TMR_BKIN input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).

17.5.2.24 ADVTMR_AF2

0x005c			TMR alternate function option register 2											ADVTMR_AF2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved														OCREFCLRSRCSEL	
Type	RO														RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				BRK2COMP3P	BRK2COMP2P	BRK2COMP1P	BRK2INP	Reserved				BRK2COMP3EN	BRK2COMP2EN	BRK2COMP1EN	BRK2INEN
Type	RO				RW	RW	RW	RW	RO				RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 17-40 TMR Alternate Function Option Register 2 Description

Field	Name	Description
31:18	Reserved	Reserved
17:16	OCREFCLRSRCSEL	ocref_clr source selection These bits select the ocref_clr input source. 00: tmr_ocref_clr0 01: tmr_ocref_clr1 ... 11: tmr_ocref_clr3 **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
15:12	Reserved	Reserved

Field	Name	Description
11	BRK2COMP3P	tmr_brk2_cmp3 input polarity This bit selects the tmr_brk2_cmp3 input sensitivity. It must be programmed together with the BRK2P polarity bit. 0: tmr_brk2_cmp3 input polarity is not inverted (active low if BRK2P = 0, active high if BRK2P = 1) 1: tmr_brk2_cmp3 input polarity is inverted (active high if BRK2P = 0, active low if BRK2P = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
10	BRK2COMP2P	tmr_brk2_cmp2 input polarity This bit selects the tmr_brk2_cmp2 input sensitivity. It must be programmed together with the BRK2P polarity bit. 0: tmr_brk2_cmp2 input polarity is not inverted (active low if BRK2P = 0, active high if BRK2P = 1) 1: tmr_brk2_cmp2 input polarity is inverted (active high if BRK2P = 0, active low if BRK2P = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
9	BRK2COMP1P	tmr_brk2_cmp1 input polarity This bit selects the tmr_brk2_cmp1 input sensitivity. It must be programmed together with the BRK2P polarity bit. 0: tmr_brk2_cmp1 input polarity is not inverted (active low if BRK2P = 0, active high if BRK2P = 1) 1: tmr_brk2_cmp1 input polarity is inverted (active high if BRK2P = 0, active low if BRK2P = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
8	BRK2INP	TMR_BKIN2 input polarity This bit selects the TMR_BKIN2 alternate function input sensitivity. It must be programmed together with the BRK2P polarity bit. 0: TMR_BKIN2 input polarity is not inverted (active low if BRK2P = 0, active high if BRK2P = 1) 1: TMR_BKIN2 input polarity is inverted (active high if BRK2P = 0, active low if BRK2P = 1)

Field	Name	Description
		NOTE: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
7:4	Reserved	Reserved
3	BRK2COMP3EN	tmr_brk2_cmp3 enable This bit enables the tmr_brk2_cmp3 for the timer's tmr_brk2 input. tmr_brk2_cmp3 output is 'ORed' with the other tmr_brk2 sources. 0: tmr_brk2_cmp3 input disabled 1: tmr_brk2_cmp3 input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
2	BRK2COMP2EN	tmr_brk2_cmp2 enable This bit enables the tmr_brk2_cmp2 for the timer's tmr_brk2 input. tmr_brk2_cmp2 output is 'ORed' with the other tmr_brk2 sources. 0: tmr_brk2_cmp2 input disabled 1: tmr_brk2_cmp2 input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
1	BRK2COMP1EN	tmr_brk2_cmp1 enable This bit enables the tmr_brk2_cmp1 for the timer's tmr_brk2 input. tmr_brk2_cmp1 output is 'ORed' with the other tmr_brk2 sources. 0: tmr_brk2_cmp1 input disabled 1: tmr_brk2_cmp1 input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).

Field	Name	Description
0	BRK2INEN	TMR_BKIN2 input enable This bit enables the TMR_BKIN2 alternate function input for the timer's tmr_brk2 input. TMR_BKIN2 input is 'ORed' with the other tmr_brk2 sources. 0: TMR_BKIN2 input disabled 1: TMR_BKIN2 input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).

17.5.2.25 ADVTMR_DMACFG

0x0060			TMR DMA control register											ADVTMR_DMACFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DBL						Reserved		DBA					
Type	RO		RW						RO		RW					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-41 TMR DMA Control Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:8	DBL	DMA burst length This 6-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the DMAR address) 000000: 1 transfer 000001: 2 transfers 000010: 3 transfers ... 010001: 18 transfers
5:0	DBA	DMA base address This 6-bit vector defines the base address for DMA transfers (when read/write access is done through the DMA address). DBA is defined as an offset starting from the address of the CTRL1 register. Example:

Field	Name	Description
		00000: CTRL1 00001: CTRL2 00010: SMCFG

17.5.2.26 ADVTMR_DMAR

0x0064			TMR DMA address for full transfer											ADVTMR_DMAR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DMAB															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DMAB															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-42 TMR DMA Address for Full Transfer Description

Field	Name	Description
31:0	DMAB	DMA register for burst accesses A read or write operation to the DMAR register accesses the register located at the address (CTRL1 address) + (DBA + DMA index) x 4 Where: CTRL1 address is the address of the control register 1; DBA is the DMA base address configured in DMACFG register; DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in DMACFG).

General-Purpose Timer (GPTMR)

This chapter describes the main features and use of the General-Purpose Timer (GPTMR).

Topics: **Page**

18.1 Introduction

The General-Purpose Timers (GPTMR) consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

The GPTMR has a wide range of applications, including the measurement of pulse lengths in input signals (input capture) and the generation of output waveforms like output compare, PWM, and complementary PWM with dead-time. Pulse lengths and waveform periods can be configured by using the timer prescaler and the system clock controller prescalers.

18.2 Features

- 16-bit or 32-bit up, down, auto-reload counter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536
- Counter is from the different clock sources as below:
  - Internal input (CLK_TMR)
  - External clock mode 0 (CH0/CH1/ITRG)
  - External clock mode 1 (ETRG)
  - Encoder mode (CH0 and CH1)
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - Single pulse mode output
- Complementary outputs with programmable dead-time (for channel 0 ~ 2 only)
- Synchronization circuit to control the timer with external signals
- Several times can be interconnected 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 the reset state or a known state
- Interrupt and DMA generation on the following events:
  - Update: Counter overflow, counter initialization (by software or internal/external trigger)
  - Different trigger event (by software events or internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
- Support incremental (quadrature) encoder for positioning
- Support hall-sensor circuitry with input XOR function for 3-phase motor control application
- Support asymmetric PWM mode with two center-aligned PWM signals to be generated with a programmable phase shift
- Support 6-step PWM generation
- Support update interrupt flag bit remapping

18.3 Functional Description

18.3.1 Block Diagram

Figure 18-1 shows the block diagram of General-Purpose Timer (GPTMR).

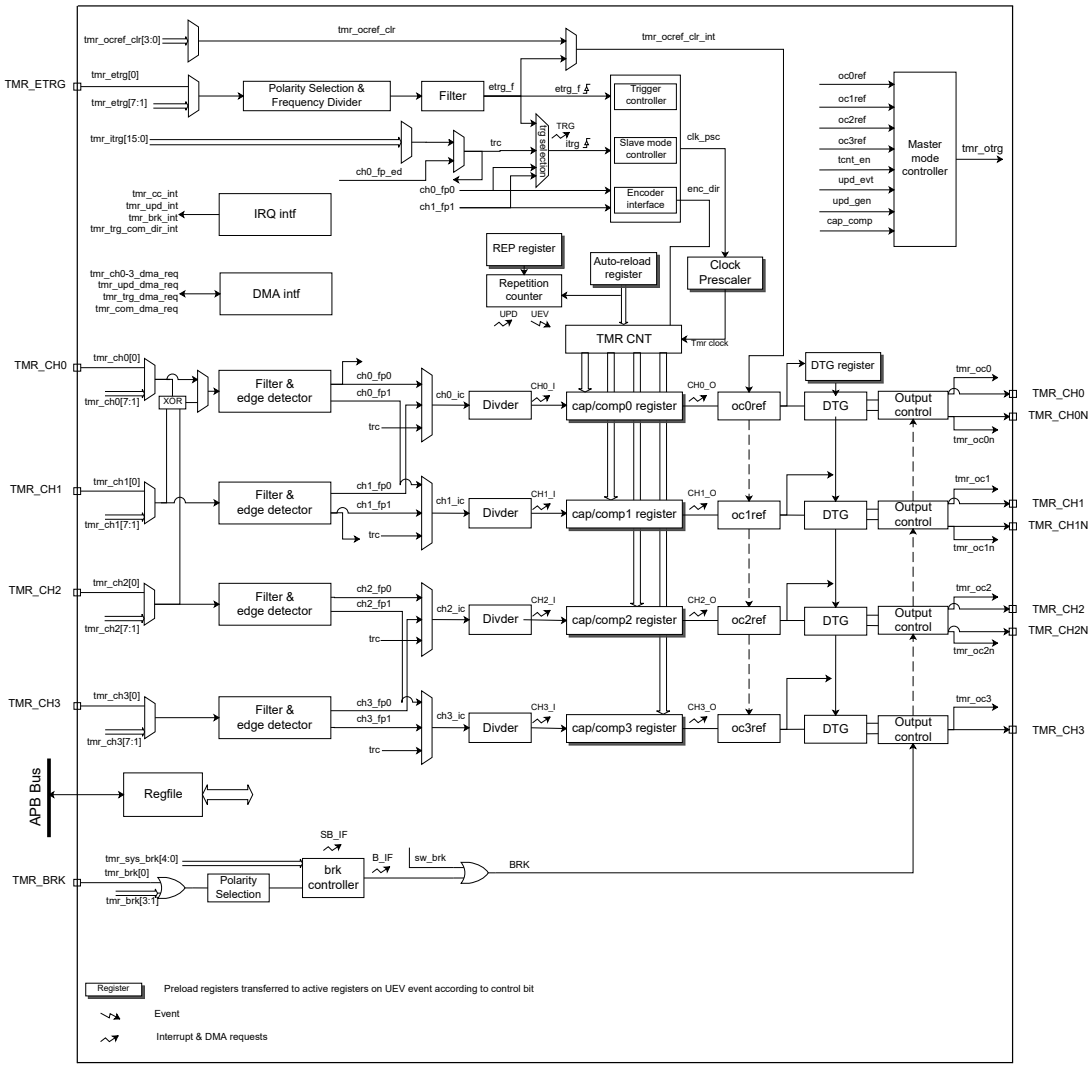


Figure 18-1 GPTMR Block Diagram

18.3.2 Pins and Internal Signals

Table 18-1 and Table 18-2 provide a summary of the inputs and outputs of the General-Purpose Timer (GPTMR).

Table 18-1 GPTMR Input/Output Pins

Pin Name	Signal Type	Description
TMR_CH0	Input/Output	Timer multi-purpose channels. 1. Each channel can be used for capture, compare or PWM. 2. TMR_CH0/1 can also be used as external clock (below 1/4 of the
TMR_CH1	Input/Output	
TMR_CH2	Input/Output	
TMR_CH3	Input/Output	

Pin Name	Signal Type	Description
		CLK_TMR clock), external trigger and quadrature encoder inputs. 3. TMR_CH0/1/2 can be used to interface with digital hall effect sensors.
TMR_CH0N	Output	Timer complementary outputs, which is derived from TMR_CHx outputs with the possibility to have dead-time insertion function.
TMR_CH1N	Output	
TMR_CH2N	Output	
TMR_ETRG	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the CLK_TMR if the tmr_etrg prescaler is used.
TMR_BRK	Input	Break input.

Table 18-2 GPTMR Internal Input/Output Signals

Internal Signal Name	I/O	Width	Bits	Description
tmr_itrg	Input	[15:0]	16	Internal trigger input bus. These inputs can be used for the slave mode controller or as an input clock (below 1/4 of the CLK_TMR clock).
tmr_etrg	Input	[7:0]	8	External trigger internal input bus. These inputs can be used as triggers, external clocks or hardware cycle-by-cycle pulse width control. These inputs can receive a clock with a frequency higher than the CLK_TMR if the tmr_etrg prescaler is used.
tmr_brk	Input	[3:0]	4	Break input bus for internal signals
tmr_sys_brk	Input	[4:0]	5	System break input. This input gathers the MCU's system level errors.
tmr_ocref_clr	Input	[3:0]	4	Timer reference output clear signals bus. These inputs can be used to clear the timer ocxref signals, typically for hardware cycle-by-cycle pulse width control. They also can be connected to the output of the internal comparators.

Internal Signal Name	I/O	Width	Bits	Description
tmr_ch0	Input	[7:0]	8	Internal timer inputs bus. These inputs can be used for capture or as an external clock (below 1/4 of the CLK_TMR clock) and for quadrature encoder signals.
tmr_ch1	Input	[7:0]	8	
tmr_ch2	Input	[7:0]	8	
tmr_ch3	Input	[7:0]	8	
tmr_otrg	Output		1	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.
tmr_cc_int	Output		1	Timer capture/compare interrupt
tmr_upd_int	Output		1	Timer update event interrupt
tmr_brk_int	Output		1	Timer break and break2 interrupt
tmr_trg_com_dir_int	Output		1	Timer trigger and commutation direction interrupt
tmr_ch0_dma_req	Output		1	Timer capture/compare channel 0/1/2/3 DMA requests
tmr_ch1_dma_req	Output		1	
tmr_ch2_dma_req	Output		1	
tmr_ch3_dma_req	Output		1	
tmr_upd_dma_req	Output		1	Timer update DMA request
tmr_trg_dma_req	Output		1	Timer trigger DMA request
tmr_com_dma_req	Output		1	Timer commutation DMA request

18.3.3 Time-Base Unit

The main block of the GPTMR is a 16-bit or 32-bit up-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, auto-reload register, and prescaler register can be both written to and read from by software, even while the counter is running.

The time-base unit consists of the Timer counter register (TCNT), Timer prescaler register (PSC), Timer auto-reload register (ARR) and Timer repetition counter register (RCR).

ARR is the value that is preloaded into the actual auto-reload register. When writing to or reading from the auto-reload register, the preload register is accessed. The content of the preload register is then transferred into the shadow register permanently or at each update event (UEV), depending on the value of the auto-reload preload enable bit (ARRSHDWEN) in the Timer CTRL1 register. The UEV occurs when the counter reaches the overflow with the down-counting setting, and if the update disable bit (UPDDIS) in the Timer CTRL1 register is set to 0. The update event can also be generated by software. The generation of the UEV is described in detail for each event generation.

The counter is clocked by the prescaler output CLK_TCNT, enabled only when the counter enable bit (TCNTEN) in the Timer CTRL1 register is set.

NOTE: The counter starts counting 1 clock cycle after setting the counter enable bit (TCNTEN) in the Timer CTRL1 register.

Prescaler

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 the Timer PSC register. It can be changed on the fly as this control register is buffered. The new prescaler ratio will be considered at the next update event (UEV).

Figure 18-2 and Figure 18-3 give some examples of the counter behavior when the prescaler ratio is changed on the fly:

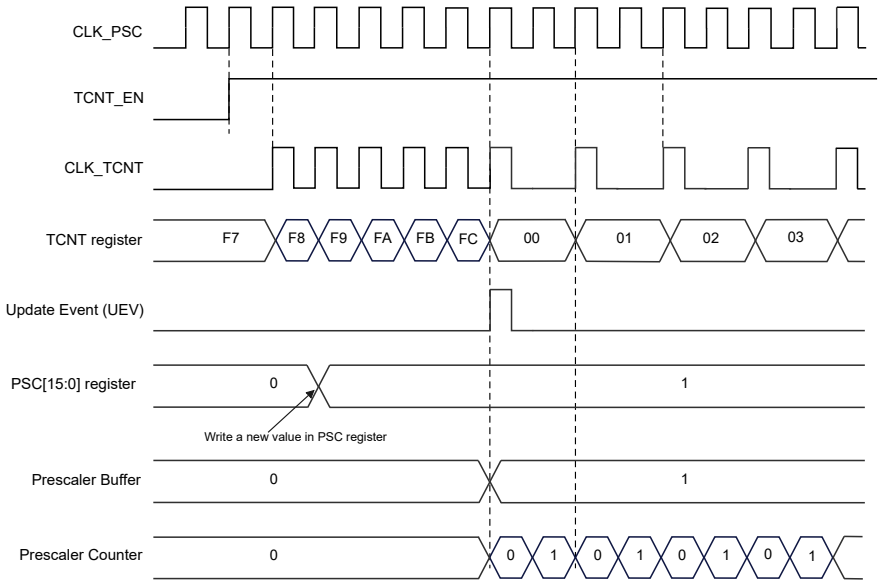


Figure 18-2 Counter Timing Diagram with Prescaler Division Change from 1 to 2

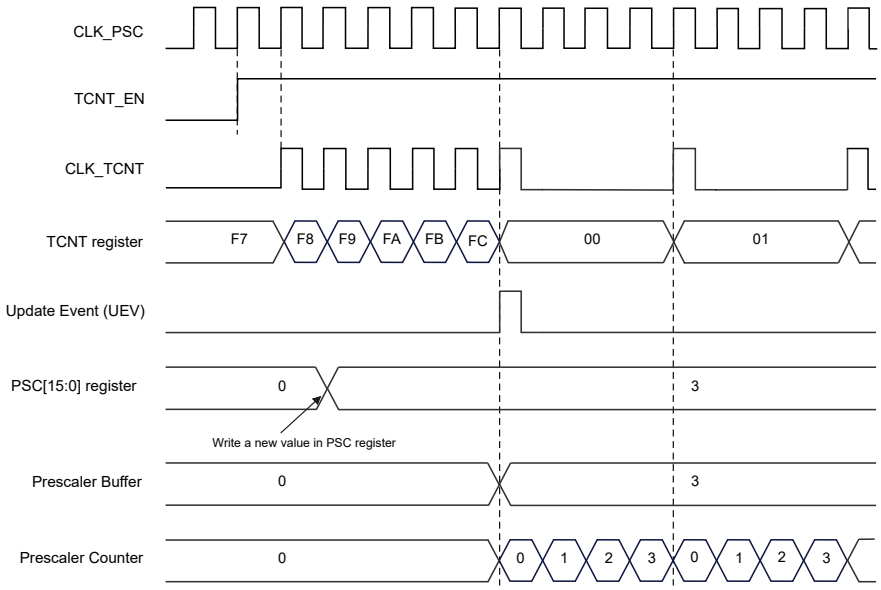


Figure 18-3 Counter Timing Diagram with Prescaler Division Change from 1 to 4

18.3.4 Counter Modes

18.3.4.1 Up-Counting

In up-counting mode, the counter counts from 0 to the auto-reload value (content in the Auto-Reload Register, ARR), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after up-counting is repeated for the number of times programmed in the Timer repetition counter register (RCR). Else the update event is generated at each counter overflow.

Setting the update generation bit (SWUPDGEN) in the Timer software event generation register (SWEVTGEN) (by software or by using the slave mode controller) also generates an update event.

Update event (UEV) can be disabled by software by setting the update disable bit (UPDDIS) to 1 in the Timer CTRL1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UPDDIS 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 UPDREQSRC bit (update request source) in Timer CTRL1 register is set, setting the SWUPDGEN bit generates an update event UEV but without setting the UPDIF 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 interrupt flag (UPDIF bit in INTSTS register) is set depending on the UPDREQSRC bit:

- The repetition counter is reloaded with the content in Timer RCR register.
- The auto-reload shadow register is updated with the preload value (Timer ARR register).
- The buffer of the prescaler is reloaded with the preload value (content in the Timer PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when Timer ARR = 0x36.

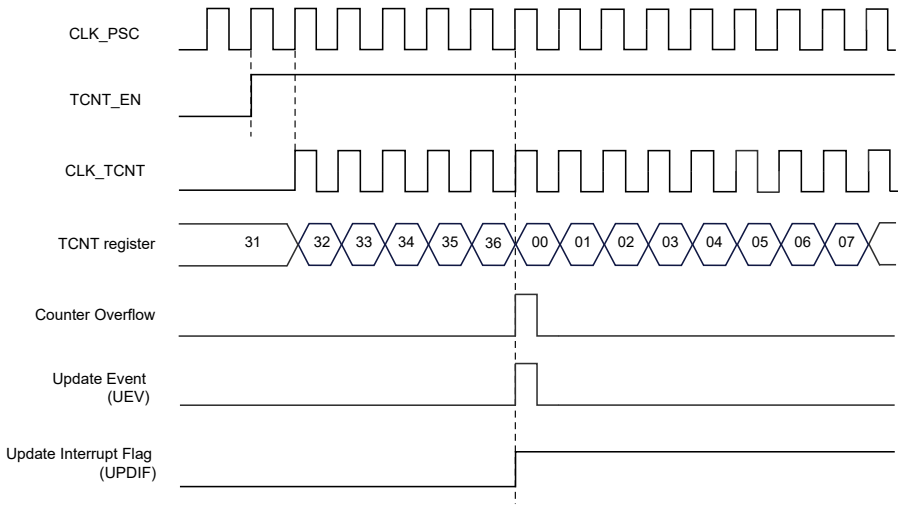


Figure 18-4 Counter Timing Diagram, Internal Clock Divided by 1

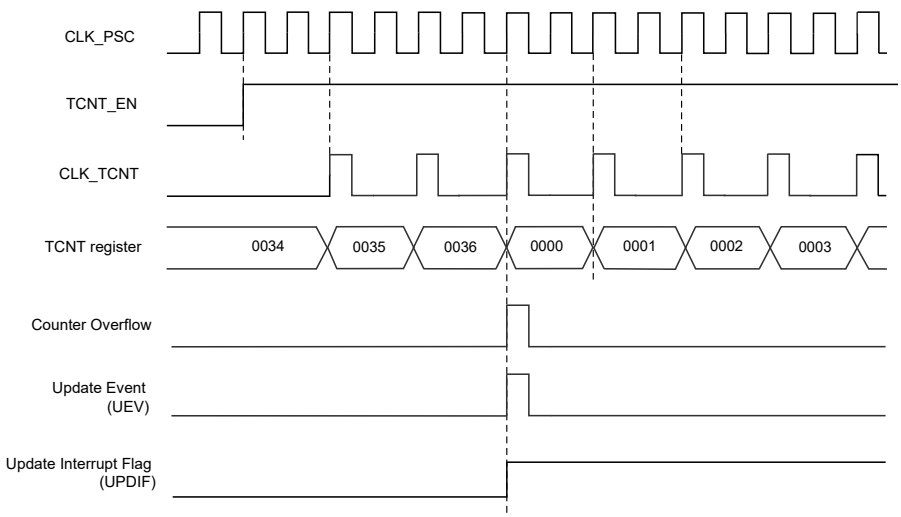


Figure 18-5 Counter Timing Diagram, Internal Clock Divided by 2

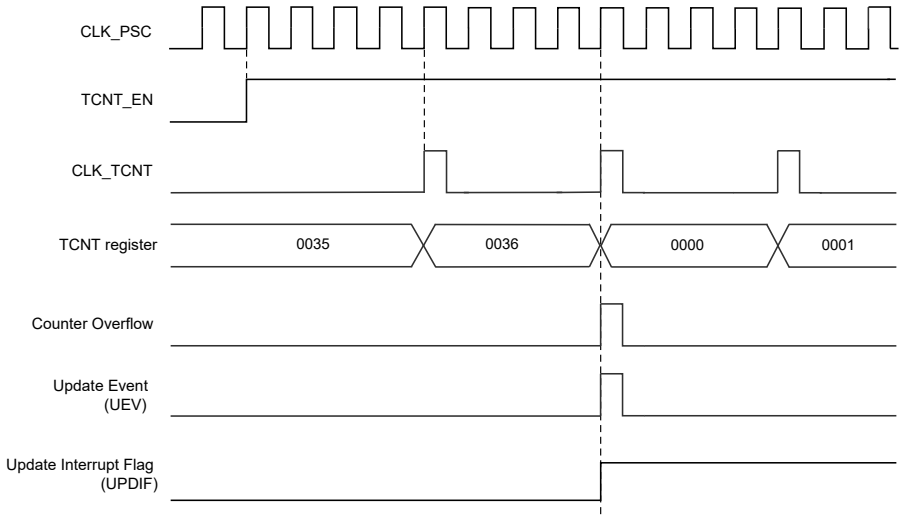


Figure 18-6 Counter Timing Diagram, Internal Clock Divided by 4

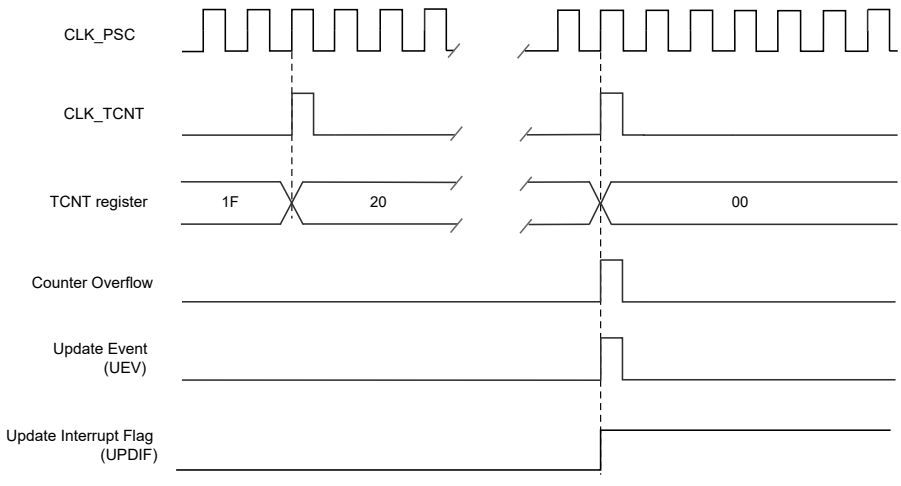


Figure 18-7 Counter Timing Diagram, Internal Clock Divided by N

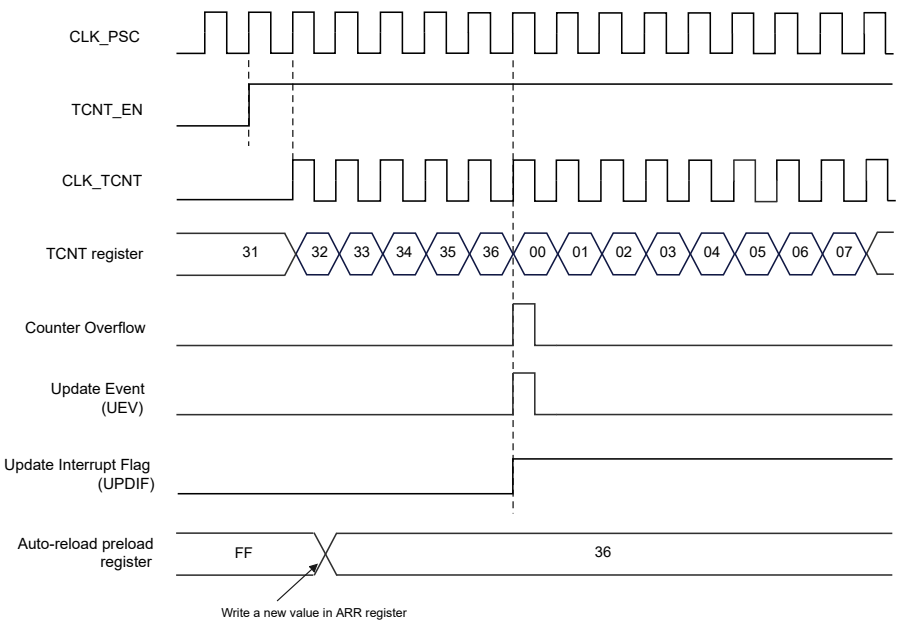


Figure 18-8 Counter Timing Diagram, Update Event When ARPE = 0 (ARR Not Preloaded)

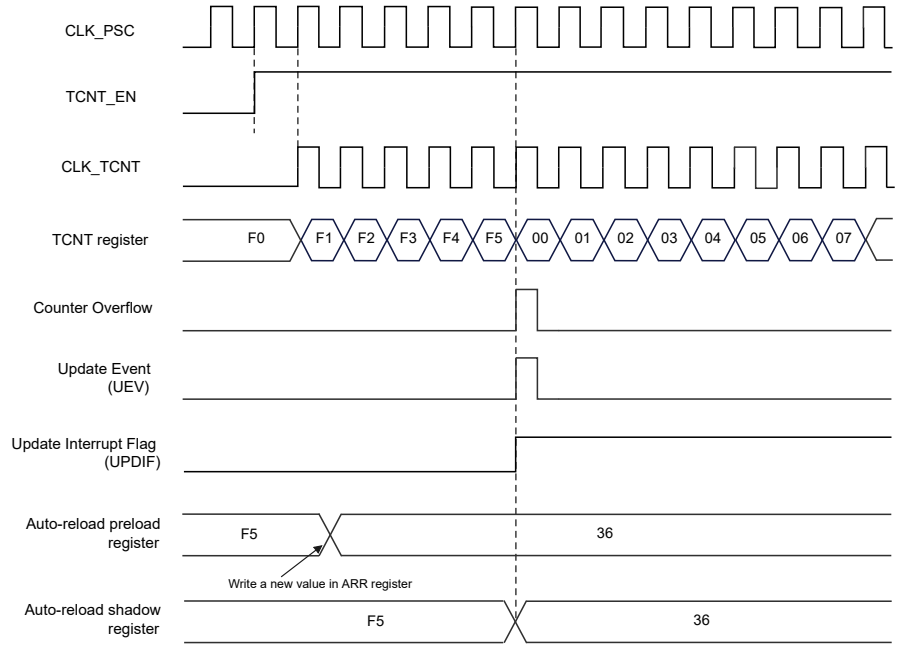


Figure 18-9 Counter Timing Diagram, Update Event When ARPE = 1 (ARR Preloaded)

18.3.4.2 Down-Counting

In down-counting mode, the counter starts from the auto-reload value stored in the Timer ARR register and decrements until it reaches 0. It then resets back to the auto-reload value and triggers a counter underflow event.

An update event (UEV) can be generated either at each counter underflow or by setting the SWUPDGEN bit in the Timer software event generation register SWEVTGEN, using either software or the slave mode controller.

The UEV event can be disabled by setting the UPDDIS bit in the Timer CTRL1 register using software. This prevents the update of shadow registers when writing new values in the preload

registers. As a result, no update event will occur until the UPDDIS bit is set to 0. However, it's important to note that while the counter restarts from the current auto-reload value, the prescaler counter restarts from 0 without any change in the prescale rate.

Additionally, if the UPDREQSRC bit in the Timer CTRL1 register is set, setting the SWUPDGEN bit will generate a UEV event without setting the UPDIF flag, which means no interrupt or DMA request will be sent. This is done to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When a UEV event occurs, all the registers are updated, and the update interrupt flag (UPDIF bit in Timer INTSTS register) is set, depending on the UPDREQSRC bit:

- The buffer of the prescaler is reloaded with the preload value (content in the Timer PSC register).
- The auto-reload active register is updated with the preload value (content in the Timer ARR register).

NOTE: The auto-reload value is updated before the counter is reloaded. This ensures that the next period is as expected.

The following figures show some examples of the counter behavior for different clock frequencies when the register ARR = 0x36.

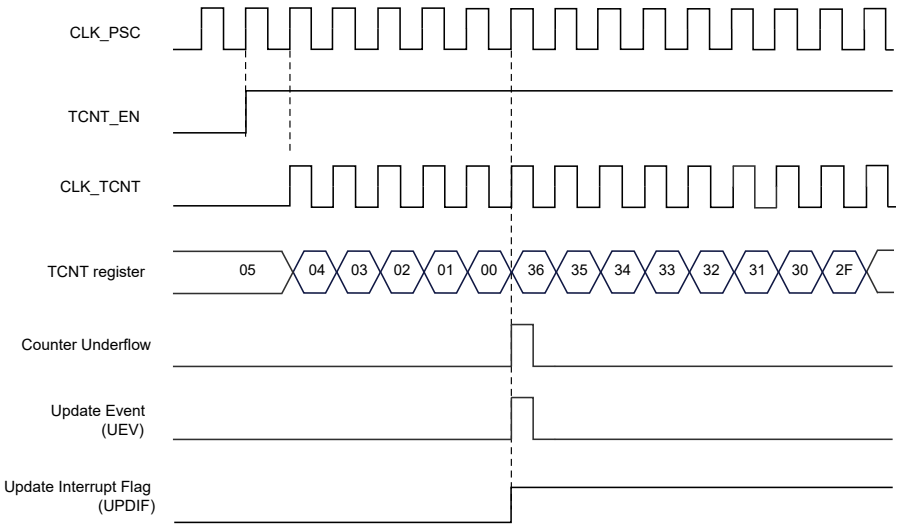


Figure 18-10 Counter Timing Diagram, Internal Clock Divided by 1

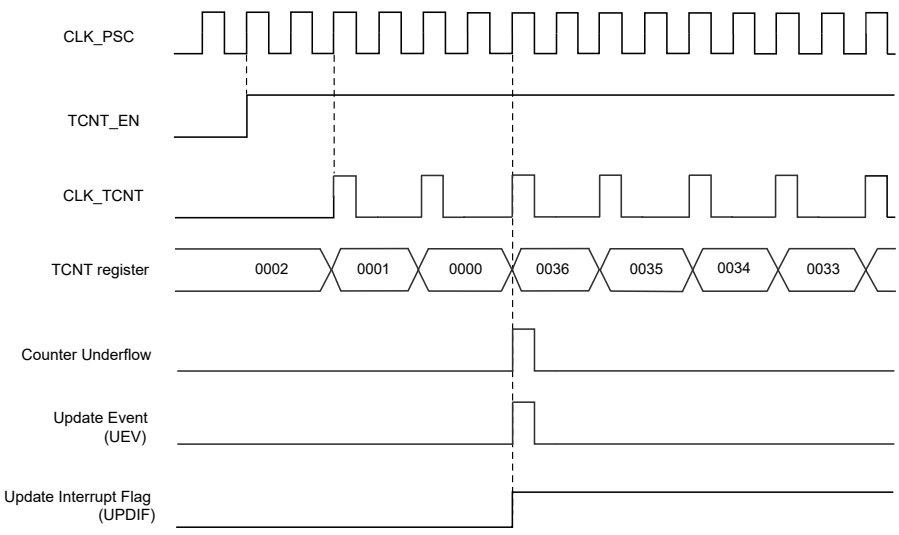


Figure 18-11 Counter Timing Diagram, Internal Clock Divided by 2

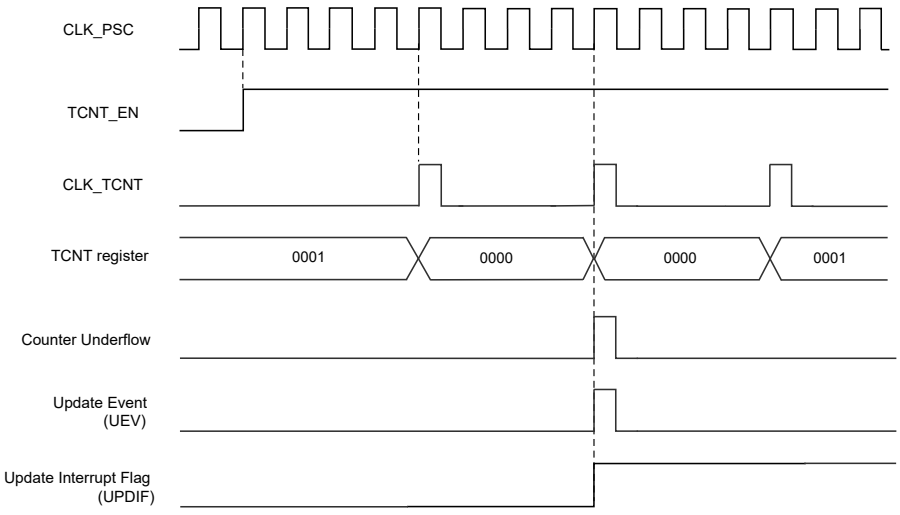


Figure 18-12 Counter Timing Diagram, Internal Clock Divided by 4

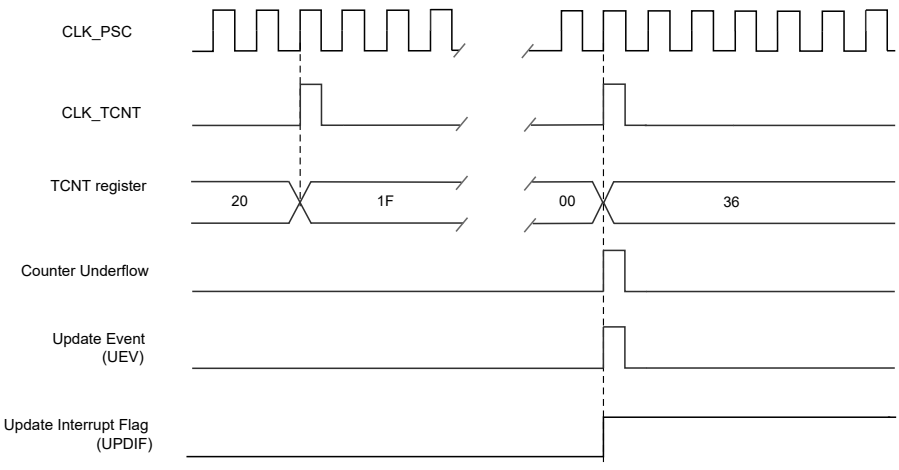


Figure 18-13 Counter Timing Diagram, Internal Clock Divided by N

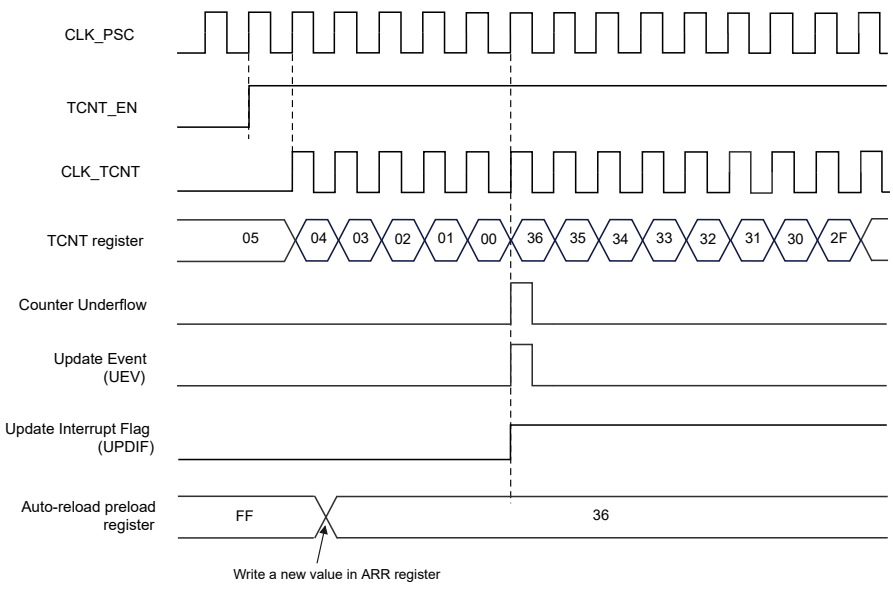


Figure 18-14 Counter Timing Diagram, Update Event

18.3.4.3 Center-Aligned Counting

In center-aligned mode, the counter starts counting from 0 and goes up to the auto-reload value minus 1. It then generates an overflow event and counts down from the auto-reload value to 1, generating an underflow event. After that, it restarts counting from 0.

The center-aligned mode is active when the TCNTALIGNMODE bits in the Timer CTRL1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when TCNTALIGNMODE is set as the following:

- The counter counts down (center-aligned mode 1, TCNTALIGNMODE = 01).
- The counter counts up (center-aligned mode 2, TCNTALIGNMODE = 10).
- The counter counts up and down (center-aligned mode 2, TCNTALIGNMODE = 11).

In this mode, the direction bit DIR in the Timer CTRL1 register cannot be written. It is updated by hardware and indicates the current direction of the counter.

The update event (UEV) can be generated at each counter overflow, and each counter underflow or by setting the SWUPDGEN bit in the Timer SWEVTGEN 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 event can be disabled by software by setting the UPDDIS bit in the Timer CTRL1 register. This avoids updating the shadow registers while writing new values in the preload registers. Then, no update event occurs until the UPDDIS bit has been written to 0. However, the counter continues counting up and down based on the current auto-reload value.

Additionally, if the UPDREQSRC bit (update request source) in the Timer CTRL1 register is set, setting the SWUPDGEN bit generates a UEV event but without setting the UPDIF 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 interrupt flag (UPDIF bit in Timer INTSTS register) is set depending on the UPDREQSRC bit:

- The buffer of the prescaler is reloaded with the preload value (content in the Timer PSC register).

- The auto-reload active register is updated with the preload value (content in the Timer ARR register).

NOTE: 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.

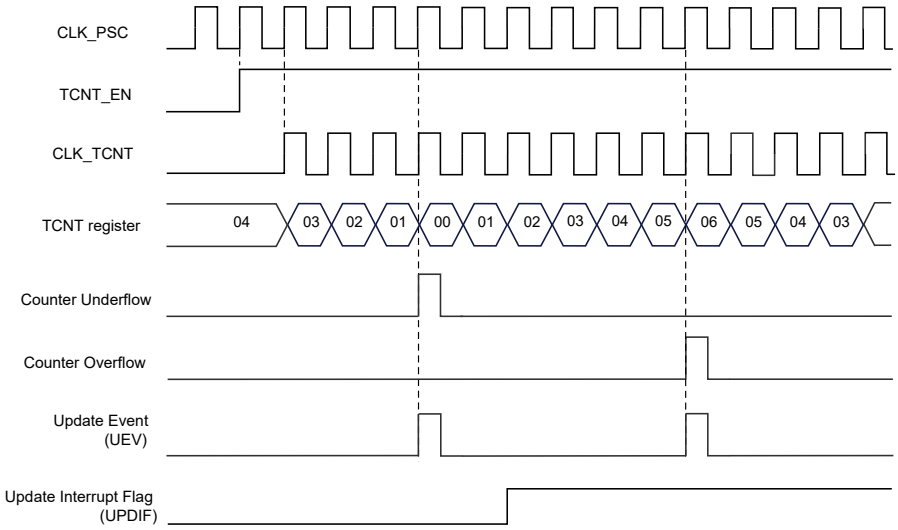


Figure 18-15 Counter Timing Diagram, Internal Clock Divided by 1, ARR = 0x6

NOTE: Center-aligned mode 1 is used.

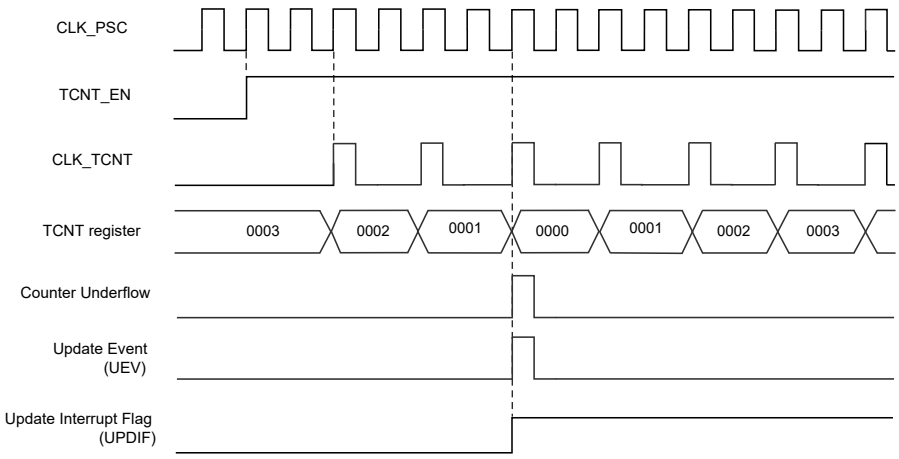


Figure 18-16 Counter Timing Diagram, Internal Clock Divided by 2

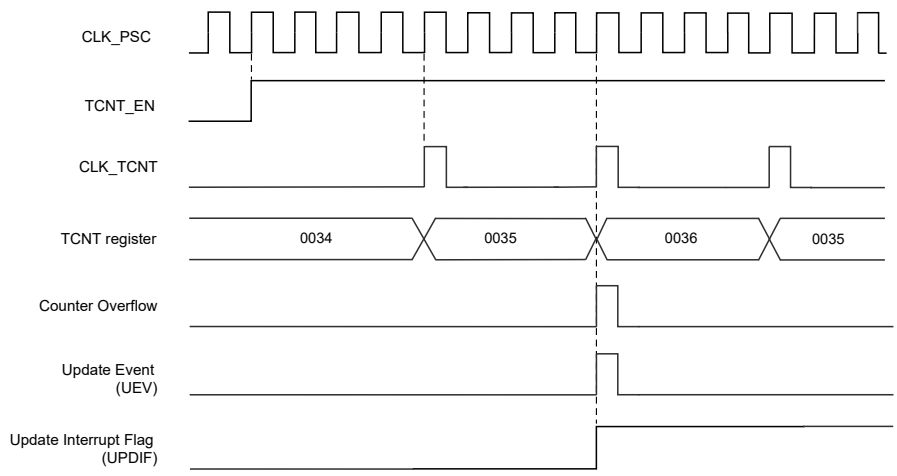


Figure 18-17 Counter Timing Diagram, Internal Clock Divided by 4, ARR = 0x36

NOTE: Center-aligned mode 2 or 3 is used with an UPDIF on overflow.

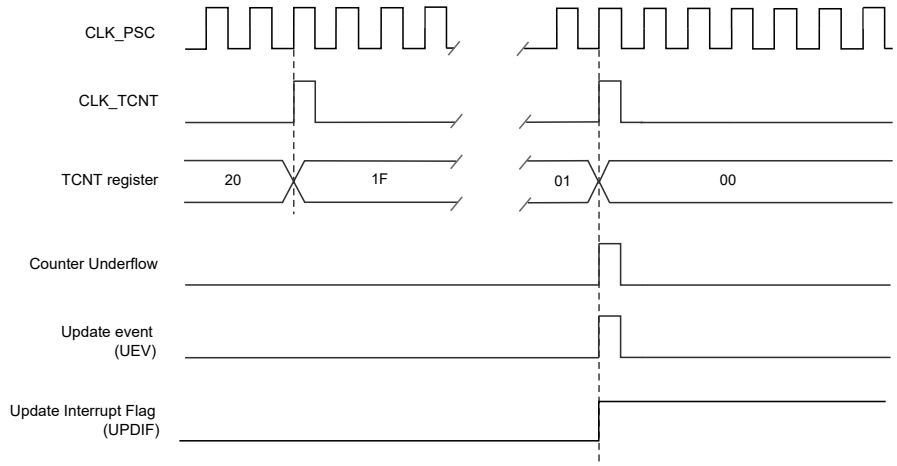


Figure 18-18 Counter Timing Diagram, Internal Clock Divided by N

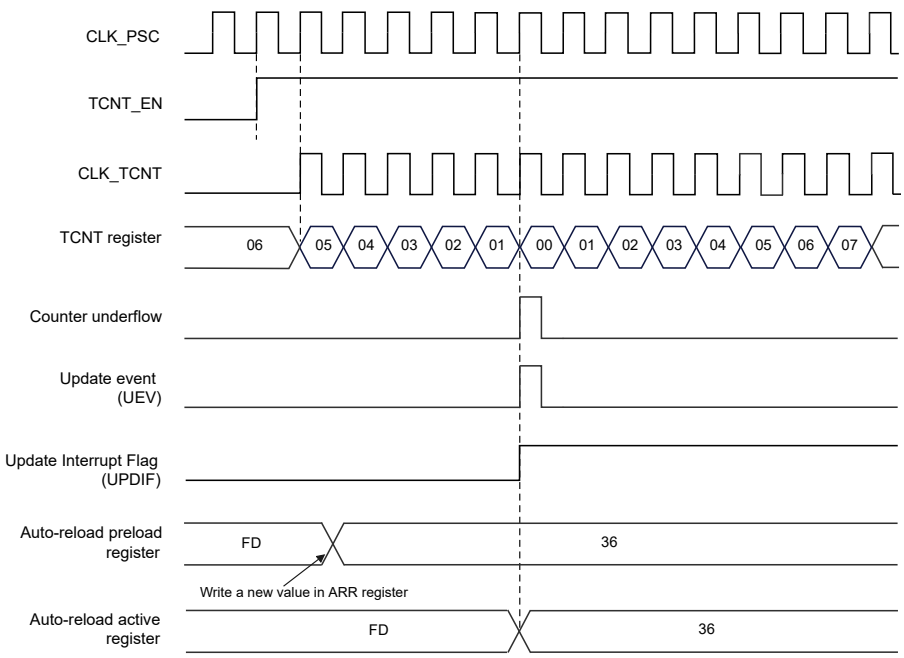


Figure 18-19 Counter Timing Diagram, UEV with ARPE = 1 (Counter Underflow)

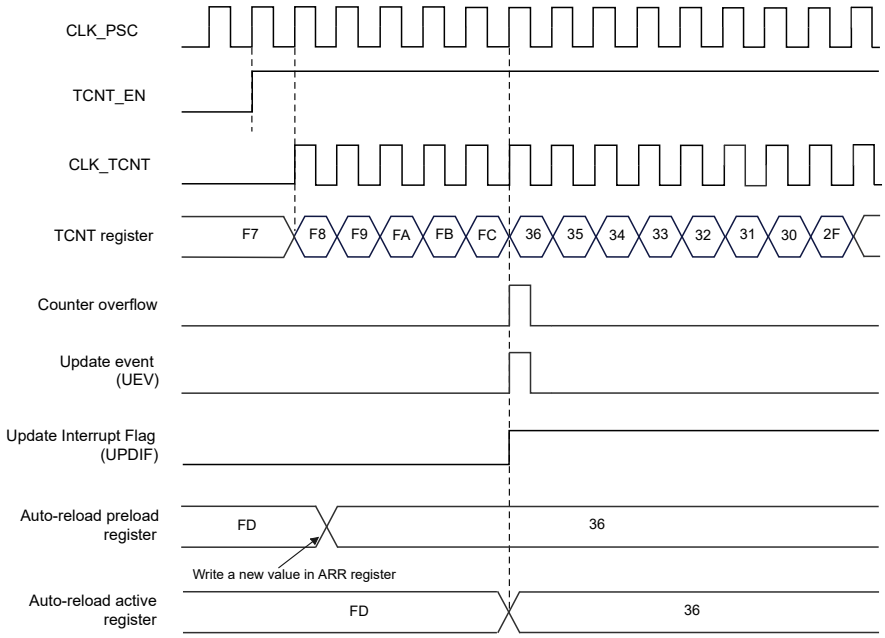


Figure 18-20 Counter Timing Diagram, UEV with ARPE = 1 (Counter Overflow)

18.3.5 Clock Selection

The counter clock can be provided by the following clock sources:

- Internal clock (CLK_TMR)
- External clock mode 0: External input pin (TMR_CH0 or TMR_CH1)
- External clock mode 1: External trigger input (TMR_ETRG)
- Internal trigger inputs (tmr_itrg): Using one timer as prescaler for another timer

Internal clock source (CLK_TMR)

If the slave mode controller is disabled (SLVMODECTRL = 0000 in the Timer SMCFG register), then the TCNTEN, DIR (in the CTRL1 register) and SWUPDGEN bits (in the SWEVTGEN register) are actual control bits and can be changed only by software (except SWUPDGEN which remains cleared automatically). As soon as the TCNTEN bit is written to 1, the prescaler is clocked by the internal clock CLK_TMR.

Figure 18-21 shows the behavior of the control circuit and the up-counter in normal mode, without prescaler.

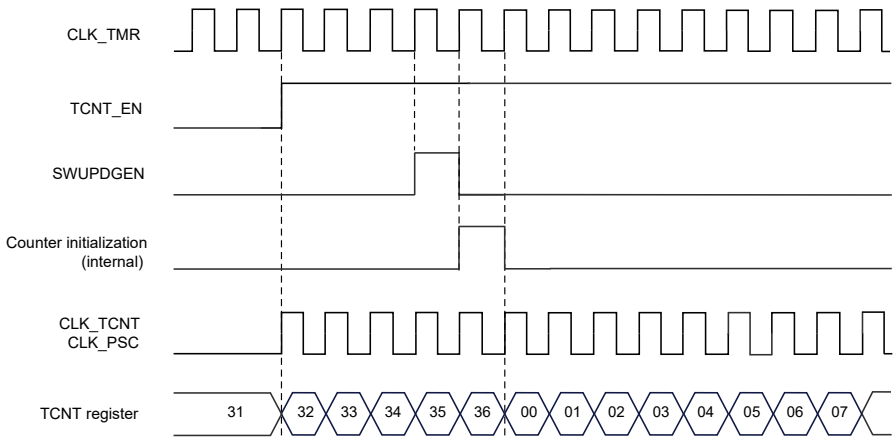


Figure 18-21 Control Circuit in Normal Mode

External clock mode 0

This mode is selected when SLVMODECTRL = 0111 in the Timer SMCFG register. It is external clock mode 0 that the counter can count at each rising edge on a selected input.

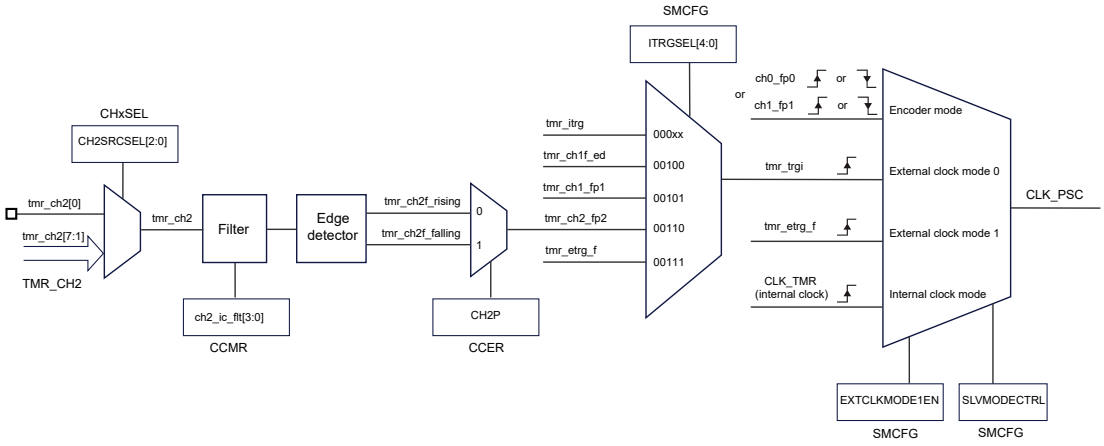


Figure 18-22 TMR_CH2 External Clock Connection Example

For example, to configure the up-counter to count in response to a rising edge on the TMR_CH2 input, follow the steps below:

1. Select the proper tmr_ch2 [7:0] source (internal or external) with the CH2SRCSEL[2:0] bits in the Timer CHxSEL register.
2. Configure channel 2 to detect rising edges on the tmr_ch2 input by writing ch2_mode_sel= 01 in the Timer CHXMODECTRL2 register.
3. Configure the input filter duration by writing the CH2ICFLT[3:0] bits in the Timer CHXMODECTRL2 register (if no filter is needed, keep CH2ICFLT = 0000).

NOTE: The capture prescaler is not used for triggering; therefore, it does not require any configuration.

4. Select rising edge polarity by writing CH2P = 0 and CH2NP = 0 in the Timer CCER register.
5. Configure the timer in external clock mode 0 by writing SLVMODECTRL = 0111 in the Timer SMCFG register.
6. Select tmr_ch2 as the input source by writing ITRGSEL = 00110 in the Timer SMCFG register.
7. Enable the counter by writing TCNTEN = 1 in the Timer CTRL1 register.

When a rising edge occurs on tmr_ch2, the counter counts once and the TRGIF flag is set.

The delay between the rising edge on tmr_ch2 and the actual clock of the counter is due to the resynchronization circuit on tmr_ch2 input.

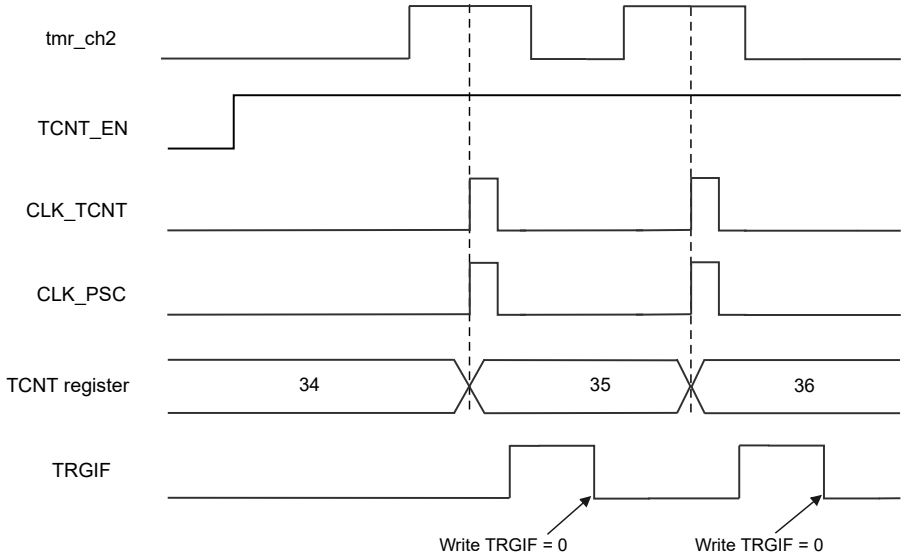


Figure 18-23 Control Circuit in External Clock Mode 0

External clock source mode 1

This mode is configured on EXTCLKMODE1EN = 1 in the Timer SMCFG register.

The counter can count at each rising or falling edge on the external trigger input tmr_etrg_f signal.

Figure 18-24 gives an overview of the external trigger input block.

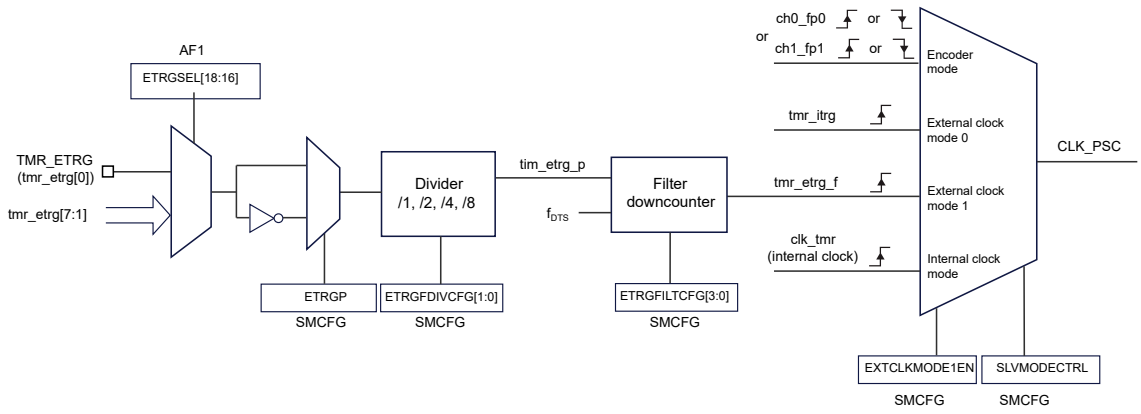


Figure 18-24 External Trigger Input Block

For example, to configure the up-counter to count each 2 rising edges on tmr_etrg, follow the steps below:

1. Select the proper tmr_etrg source (internal or external) with the ETRGSEL[18:16] bits in the Timer AF1 register.
2. Write ETRGFILTCFG[3:0] = 0000 in the Timer SMCFG register.
3. Set the prescaler by writing ETRGFDIVCFG[1:0] = 01 in the Timer SMCFG register since no filter is required in this example.
4. Select rising edge detection on the tmr_etrg by writing ETRGP = 0 in the Timer SMCFG register.
5. Enable external clock mode 1 by writing EXTCLKMODE1EN = 1 in the Timer SMCFG register.
6. Enable the counter by writing TCNTEN = 1 in the Timer CTRL1 register.

The counter counts once each 2 tmr_etrg rising edges.

The delay between the rising edge on tmr_etrg and the actual clock of the counter is due to the resynchronization circuit on the tmr_etrg signal. Therefore, the maximum frequency that can be correctly captured by the counter is at most ¼ of CLK_TMR frequency.

If the tmr_etrg signal is faster, it is recommended to apply a division of the external signal using a proper ETRGFDIVCFG prescaler setting.

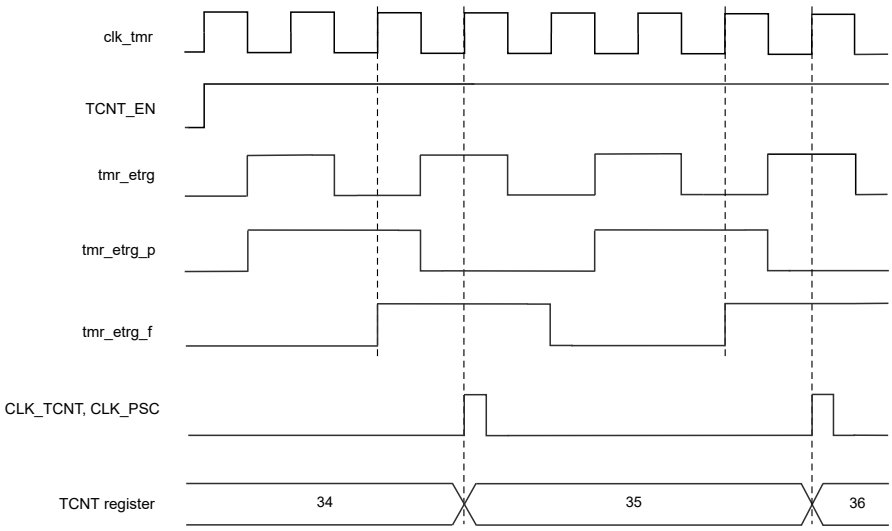


Figure 18-25 Control Circuit in External Clock Mode 1

18.3.6 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 following figure gives an overview of one capture/compare channel.

The input stage samples the corresponding TMR_CHx input to generate a filtered signal tmr_chxf. Then, an edge detector with polarity selection generates a signal (tmr_chx_fpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the input capture prescaler register.

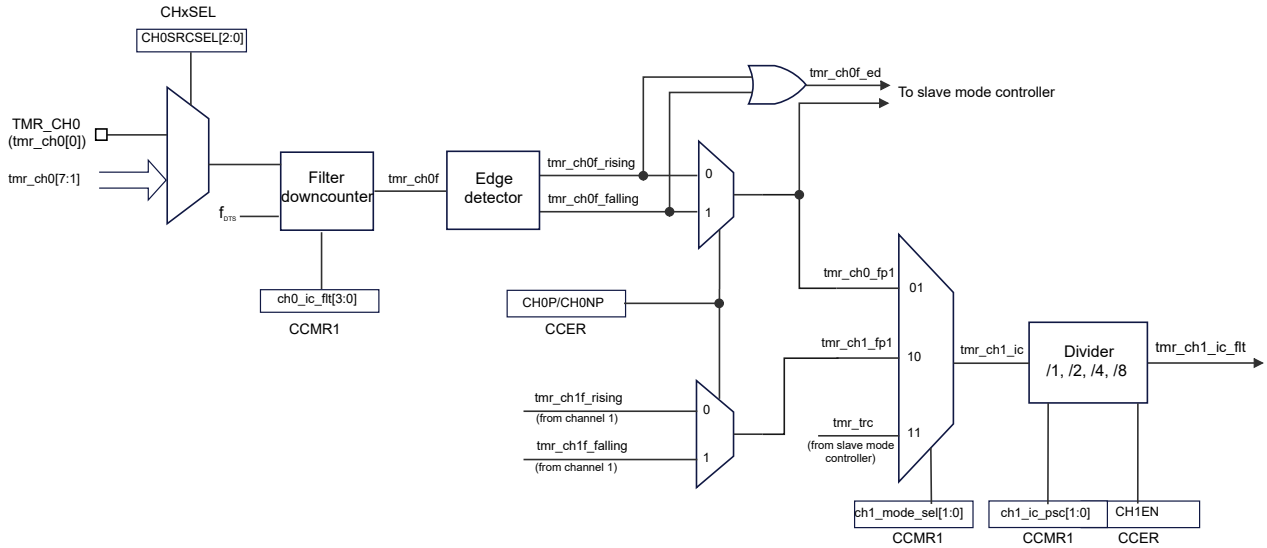


Figure 18-26 Capture/Compare Channel (Example: Channel 1 Input Stage)

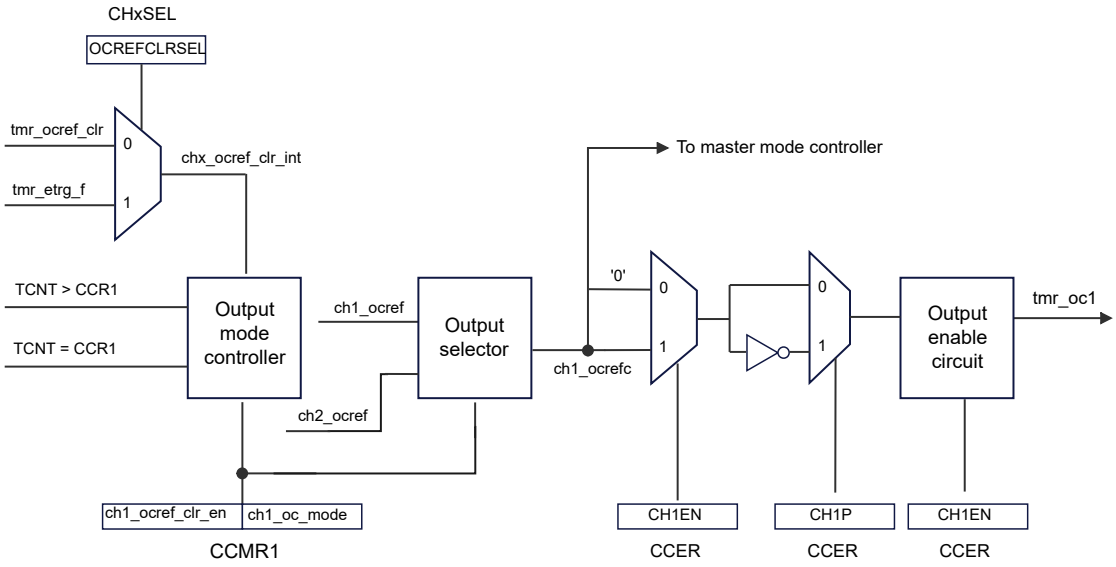


Figure 18-27 Output Stage of Capture/Compare Channel (Channel 1)

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.

18.3.7 Input Capture Mode

In input capture mode, the capture/compare registers (CCRx) are used to latch the value of the counter after a transition detected by the corresponding chx_ic signal. When a capture occurs, the corresponding CHxIF flag (Timer INTSTS register) is set, and an interrupt or a DMA request can be sent if enabled. If a capture occurs while the CHxIF flag is already high, then the over-capture flag CHxOF (Timer INTSTS register) is set. CHxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the Timer CCRx register. CHxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in Timer CCR1 when tmr_ch1 input rises.

Procedure

1. Select the proper tmr_chx_in[7:0] source (internal or external) with the CH1SRCSEL[2:0] bits in the Timer CHxSEL register.
2. Select the active input: Timer CCR1 must be linked to the tmr_ch1 input, so write the CH1MODESEL bits to 01 in the Timer CHXMODECTRL1 register. When CH1MODESEL differs from 00, the channel is configured in input, and the Timer CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tmr_chx (CHxICFLT bits in the Timer CCMRx register).

   For example, if the input signal is unstable for at least 5 internal clock cycles during toggling, program a filter duration longer than 5 clock cycles. To validate a transition on tmr_ch1, it is necessary to detect 8 consecutive samples with the new level (sampled at f_{DTS} frequency) and set the CH1ICFLT bits to 0011 in the Timer CHXMODECTRL1 register.
4. Select the edge of the active transition on the tmr_ch1 channel by writing the CH1P and CH1NP bits to 000 in the Timer CCER register (in this case, it's the rising edge).
5. Program the input prescaler.

   In this example, disable the prescaler (set CH1ICPSC bits to 00 in the Timer CHXMODECTRL1 register) to perform the capture at each valid transition.
6. Enable capture from the counter into the capture register by setting the CH1EN bit in the Timer CCER register.
7. If necessary, enable the related interrupt request by setting the CH1IE bit in the Timer DMAINTEN register and/or the DMA request by setting the CH1DE bit in the Timer DMAINTEN register.

When an input capture occurs:

- The Timer CCR1 register gets the value of the counter on the active transition.
- CH1IF flag is set (interrupt flag). CH1OF is also set if at least two consecutive captures occurred, whereas the flag was not cleared.
- An interrupt is generated depending on the CH1IE bit.
- A DMA request is generated depending on the CH1DE bit.

To handle the over-capture, it is recommended to read the data before the over-capture flag. This is to avoid missing an over-capture, 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 SWCHxCCGEN bit in the Timer SWEVTGEN register.

18.3.8 PWM Input Mode

This mode allows the measurement of the period and the duty cycle of a PWM signal connected to a single tmr_chx input:

- The CCR0 register holds the period value (interval between two consecutive rising edges).
- The CCR1 register holds the pulse width (interval between two consecutive rising and falling edges).

This mode is a particular case of input capture mode. The setup process is similar, but with the following distinctions:

- Two ICx signals are mapped on the same tmr_chx input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two ch0_fp0 and ch1_fp1 signals is selected as trigger input, and the slave mode controller is configured in reset mode.

To measure the period and pulse width of a PWM signal applied on tmr_ch0, follow these steps:

1. Select the proper tmr_chx[7:0] source (internal or external) with the CH0SRCSEL[2:0] bits in the Timer CHxSEL register.
2. Select the active input for Timer CCR0: Write the CH0MODESEL bits to 01 in the Timer CHXMODECTRL1 register (tmr_ch0 selected).
3. Select the active polarity for tmr_ch0fp0 (used both for capture in Timer CCR0 and counter clear): Write the CH0P to 0 and the CH0NP bit to 0 (active on rising edge).
4. Select the active input for Timer CCR1: Write the CH1MODESEL bits to 10 in the Timer CHXMODECTRL1 register (tmr_ch0 selected).
5. Select the active polarity for tmr_ch0fp1 (used for capture in Timer CCR1): Write the CH1P bit to 1 and the CH1NP bit to 0 (active on falling edge).
6. Select the valid trigger input: Write the ITRGSEL bits to 00101 in the Timer SMCFG register (tmr_ch0fp0 selected).
7. Configure the slave mode controller in reset mode: Write the SLVMODECTRL bits to 0100 in the Timer SMCFG register.
8. Enable the captures: Write the CH0EN and CH1EN bits to 1 in the Timer CCER register.

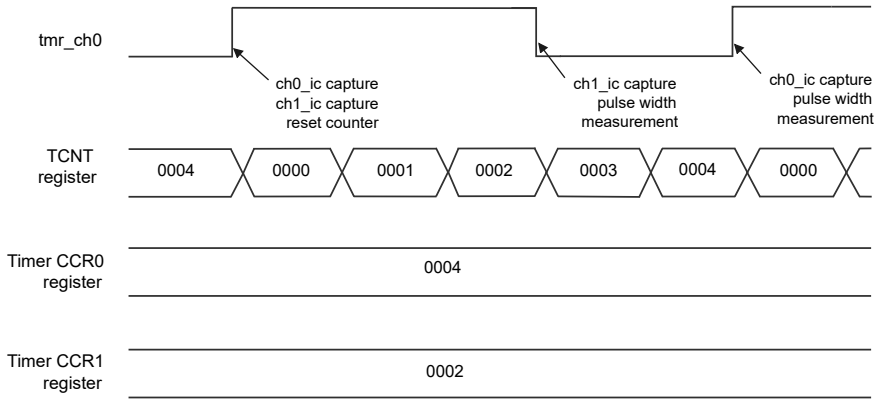


Figure 18-28 PWM Input Mode Timing

18.3.9 Forced Output Mode

In output mode (CHxMODESEL bits = 00 in the Timer CCMRx register), the software can directly control the activation or deactivation of each output compare signal (ocxref/tmr_ocx), regardless of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/tmr_ocx) to its active level, it simply needs to write 101 in the CHxOCMODE bits in the corresponding Timer CCMRx register. This will force ocxref to the high level (ocxref is always active high), and tmr_ocx will have the opposite value of the CHxP polarity bit.

As an example, when CHxP is set to 0 (meaning tmr_ocx is in an active high state), it results in tmr_ocx being driven to a high level. The ocxref signal can be made to go low by setting the CHxOCMODE bits to 0100 in the Timer CCMRx register.

However, the comparison between the Timer 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 Output Compare Mode.

18.3.10 Output Compare Mode

This function is used to control an output waveform or to indicate when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (CHxOCMODE bits in the Timer CCMRx register) and the output polarity (CHxP bit in the Timer CCER register). The output pin can keep its level (CHxOCMODE = 0000), be set active (CHxOCMODE = 0001), be set inactive (CHxOCMODE = 0010) or can toggle (CHxOCMODE = 0011) on the match.
- Sets a flag in the interrupt status register (CHxIF bit in the Timer INTSTS register).
- Generates an interrupt if the corresponding interrupt mask is set (CHxIE bit in the Timer DMAINTEN register).
- Sends a DMA request if the corresponding enable bit is set (CHxDE bit in the Timer DMAINTEN register, CCDMAREQSRC bit in the Timer CTRL2 register for the DMA request selection).

The Timer CCRx registers can be programmed with or without preload registers using the CHxCCRSHDWEN bit in the Timer CCMRx register.

In output compare mode, the update event (UEV) does not affect tmr_ocxref and tmr_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in single-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the Timer ARR and Timer CCRx registers.
3. Set the CHxIE and/or CHxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a. Write CHxOCMODE = 0011 to toggle tmr_ocx output pin when TCNT matches CCRx.
   b. Write CHxCCRSHDWEN = 0 to disable the preload register.
   c. Write CHxP = 0 to select active high polarity.
   d. Write CHxE = 1 to enable the output.
5. Enable the counter by setting the TCNTEN bit in the Timer CTRL1 register.

The Timer CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (CHxCCRSHDWEN = 0, else Timer CCRx shadow register is updated only at the next UEV event.

An example is given in Figure 18-29.

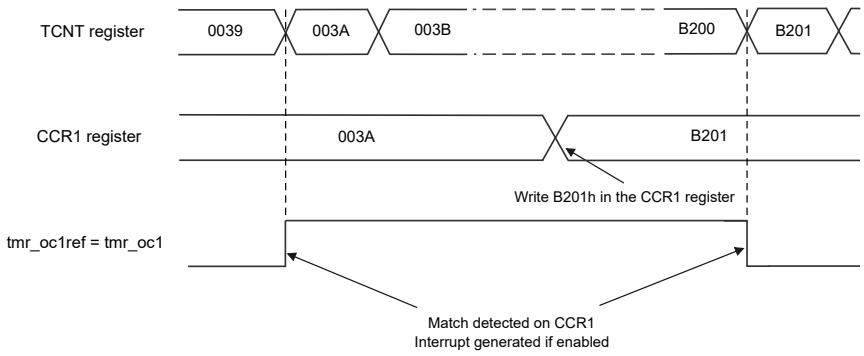


Figure 18-29 Output Compare Mode, Toggle on tmr_oc1

18.3.11 PWM Mode

The Pulse Width Modulation (PWM) mode allows the generation of a signal with a frequency determined by the value of the Timer ARR register, and a duty cycle determined by the value of the Timer CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tmr_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the CHxOCMODE bits in the Timer CCMRx register. The corresponding preload register must be enabled by setting the CHxCCRSHDWEN bit in the Timer CCMRx register, and eventually the auto-reload preload register (in up-counting or center-aligned modes) by setting the ARRSHDWEN bit in the Timer CTRL1register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the SWUPDGEN bit in the Timer SWEVTGEN register.

The tmr_ocx polarity is software programmable using the CHxP bit in the Timer CCER register. It can be programmed as active high or active low. tmr_ocx output is enabled by the CH1EN bit in the Timer CCER register. Refer to the Timer CCERx register description for more details.

In PWM mode 1 or 2, Timer TCNT and Timer CCRx are always compared to determine whether CCRx ≤ TCNT or TCNT ≤ CCRx (depending on the direction of the counter). The tmr_ocref_clr can be cleared by an external event through the tmr_etrg or the tmr_ocref_clr signals.

In this case, the tmr_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (CHxOCMODE bits in Timer CCMRx register) switches from the “frozen” configuration (no comparison, CHxOCMODE = 000) to one of the PWM modes (CHxOCMODE = 0110 or 0111), this forces the PWM by software while the timer is running.

The timer can generate PWM in edge-aligned mode or center-aligned mode depending on the TCNTALIGNMODE[6:5] bits in the Timer CTRL1 register.

Up-Counting configuration

When the DIR bit in the Timer CTRL1 register is low, the up-counting mode is active. Refer to Up-Counting for more details.

In the following example, PWM mode 1 is considered. The reference PWM signal, ocxref, remains high as long as TCNT is less than CCRx. However, it becomes low when TCNT is equal to or exceeds CCRx. If the compare value in the Timer CCRx is greater than the auto-reload value stored in the ARR register, ocxref is maintained at 1. Conversely, if the compare value is 0, ocxref is maintained at 0.

Figure 18-30 shows some edge-aligned PWM waveforms in an example where Timer ARR = 8.

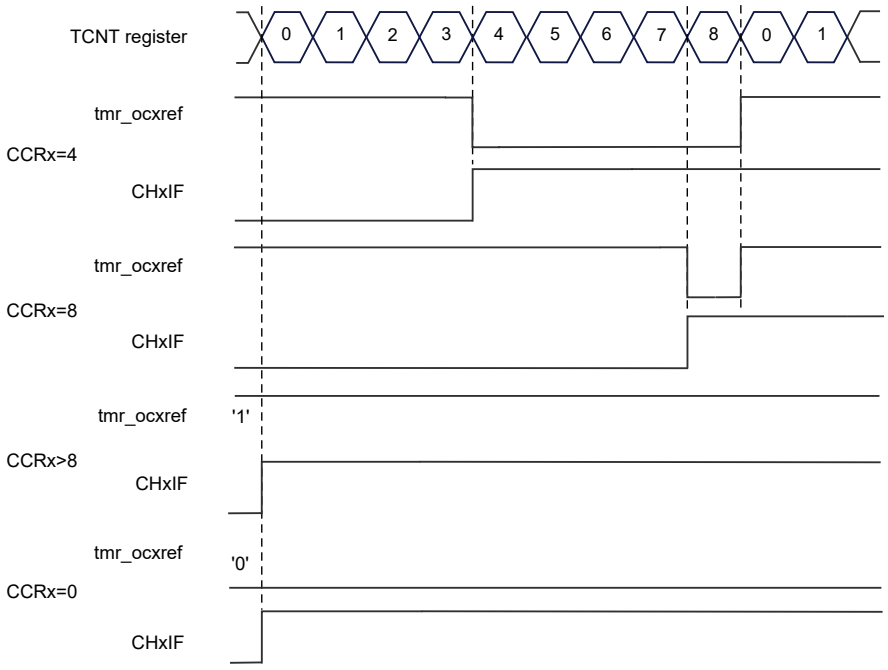


Figure 18-30 Edge-Aligned PWM Waveforms (ARR = 8)

Down-Counting Configuration

When the DIR bit in the Timer CTRL1 register is high, the down-counting mode is active. Refer to Down-Counting for more details.

In PWM mode 1, the reference signal ocxref is low as long as TCNT is greater than CCRx. However, it becomes high when TCNT is equal to or less than CCRx. If the compare value in the Timer CCRx is greater than the auto-reload value stored in the Timer ARR register, ocxref is maintained at 1. It is not possible to have a PWM duty cycle of 0% in this mode.

PWM Center-Aligned Mode

When the TCNTALIGNMODE bits in the Timer CTRL1 register have a value other than 00, the center-aligned mode is active. Different configurations of these bits have the same effect on the ocxref/tmr_ocx signals. The compare flag is set when the counter counts up, counts down, or both, depending on the configuration of the TCNTALIGNMODE bits. The direction bit (DIR) in the Timer CTRL1 register is updated by hardware and should not be modified by software. For more detailed information, please refer to the section on Center-Aligned Counting.

Figure 18-31 shows some center-aligned PWM waveforms in an example where:

- Timer register 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 TCNTALIGNMODE = 01 in Timer CTRL1 register.

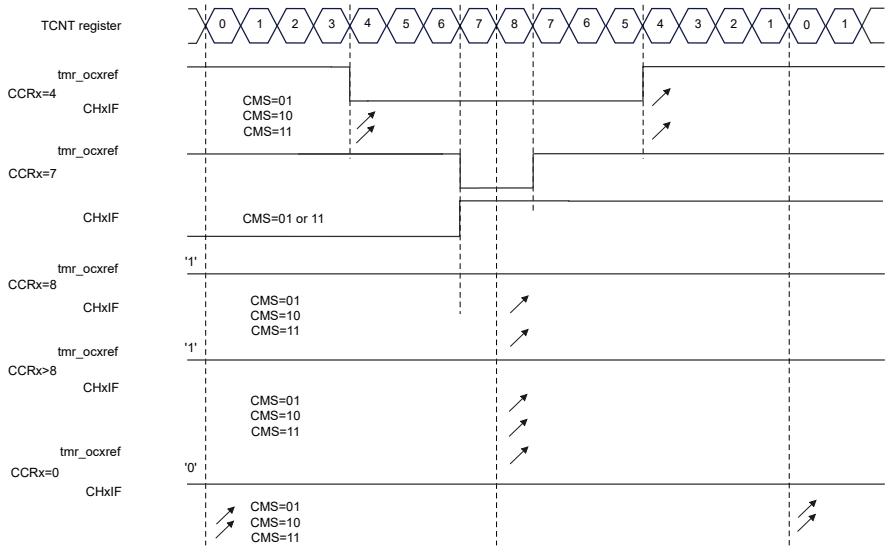


Figure 18-31 Center-Aligned PWM Waveforms (ARR = 8)

Here are some suggestions for 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 Timer CTRL1 register. Moreover, the DIR and TCNTALIGNMODE 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 a value greater than the auto-reload value is written in the counter (TCNT> ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the Timer ARR value is written 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 SWUPDGEN bit in the Timer SWEVTGEN register) just before starting the counter and not to write the counter while it is running.

18.3.12 Asymmetric PWM Mode

The asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the Timer ARR register, the duty cycle and the phase shift are determined by a pair of Timer CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle:

- tmr_oc0refc (or tmr_oc1refc) is controlled by the registers CCR0 and CCR1
- tmr_oc2refc (or tmr_oc3refc) is controlled by the registers CCR2 and CCR3

This mode can be selected independently on two channels (one tmr_ocx output per pair of CCR registers) by writing 1110 (asymmetric PWM mode 1) or 1111 (asymmetric PWM mode 2) in the CHxOCMODE bits in the Timer CCMRx register.

NOTE: To ensure compatibility, the CHxOCMODE[3:0] bit field is divided into two parts, where the most significant bit is not adjacent to the three least significant bits.

When a given channel is used as an asymmetric PWM channel, its secondary channel can also be used. For instance, if a tmr_oc0refc signal is generated on channel 0 (asymmetric PWM mode 1), it is possible to output either the ch1_ocref signal on channel 1 or a tmr_oc1refc signal resulting from asymmetric PWM mode 2.

Figure 18-32 shows an example of signals that can be generated using asymmetric PWM mode (channels 0 to 3 are configured in asymmetric PWM mode 2).

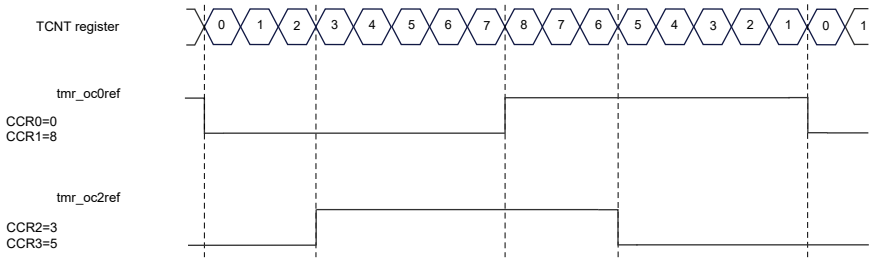


Figure 18-32 Generation of 2 Phase-shifted PWM Signals with 50% Duty Cycle

18.3.13 Combined PWM Mode

The combined PWM mode allows two edges or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the Timer ARR register, the duty cycle and delay are determined by the two Timer CCRx registers. The resulting signals, ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tmr_oc0refc (or tmr_oc1refc) is controlled by the registers CCR0 and CCR1
- tmr_oc2refc (or tmr_oc3refc) is controlled by the registers CCR2 and CCR3

This mode can be selected independently on two channels (one tmr_ocx output per pair of CCR registers) by writing 1100 (combined PWM mode 1) or 1101 (combined PWM mode 2) in the CHxOCMODE bits in the Timer CCMRx register.

When a given channel is used as a combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in combined PWM mode 1 and the other in combined PWM mode 2).

NOTE: To ensure compatibility, the CHxOCMODE[3:0] bit field is divided into two parts, where the most significant bit is not adjacent to the three least significant bits.

Figure 18-33 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 0 is configured in combined PWM mode 2
- Channel 1 is configured in PWM mode 1
- Channel 2 is configured in combined PWM mode 2
- Channel 3 is configured in PWM mode 1

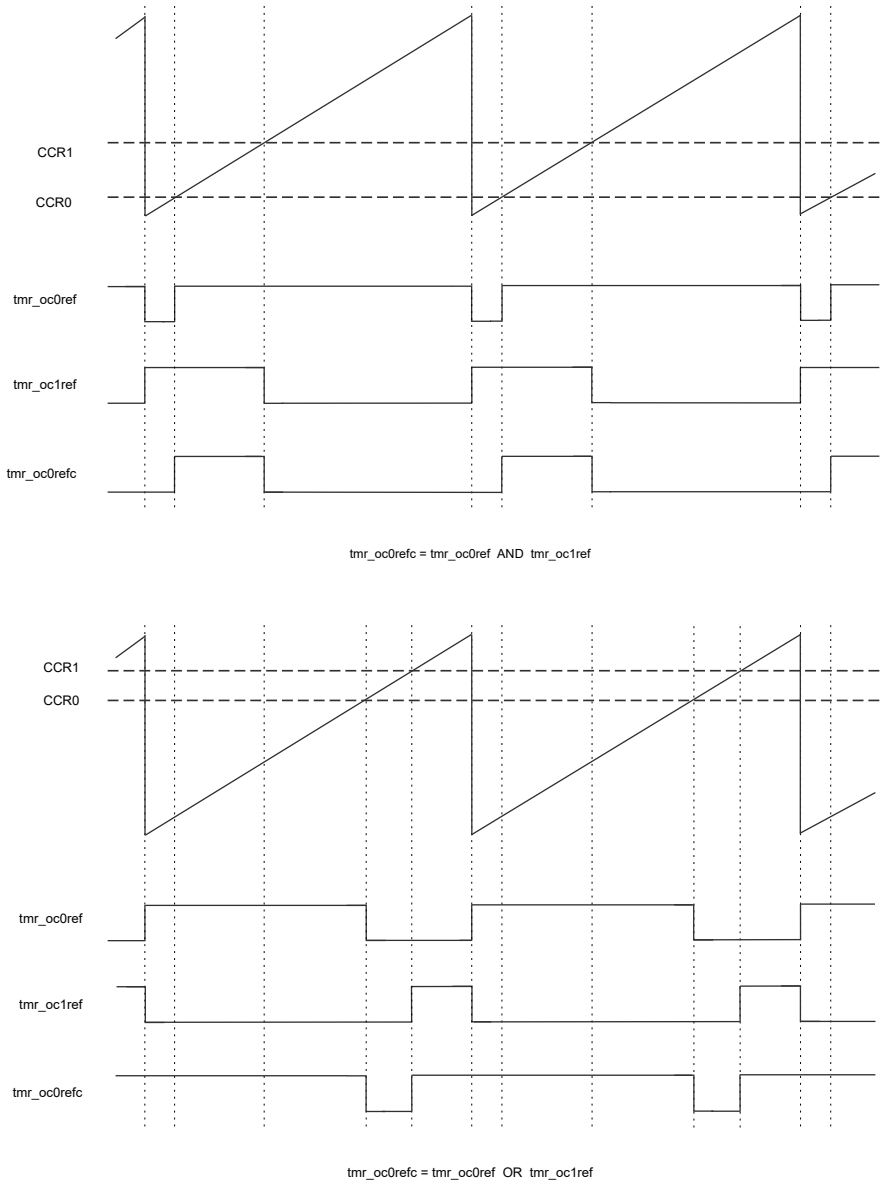


Figure 18-33 Combined PWM Mode on Channels 1 and 3

18.3.14 Clearing the OCxREF Signal on an External Event

The ocxref signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (CH0OCREFCLREN enable bit in the corresponding Timer CCMRx register set to 1). The ocxref remains low until the next update event (UEV) occurs. This function can only be used in output compare and PWM modes. It does not work in forced mode.

If the OCREF clear selection feature is implemented, the tmr_ocref_clr_int can be selected between the tmr_ocref_clr input and the tmr_etrg_f input (tmr_etrg after the filter) by configuring the OCREFCLRSEL bit in the Timer SMCFG register. The tmr_ocref_clr input can be selected among several tmr_ocref_clr[3:0] inputs, using the OCREFCLRSRCSEL[1:0] bit field in the Timer AF2 register, as shown in Figure 18-34.

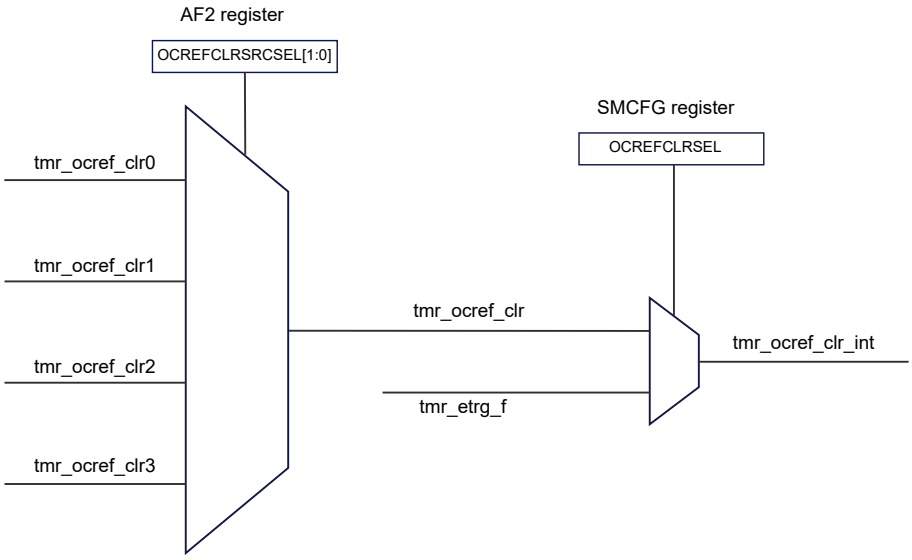


Figure 18-34 Tmr_ocref_clr Input Selection Multiplexer

If the OCREF clear selection feature is not implemented, the tmr_ocref_clr_int input is directly connected to the tmr_etrg input.

For example, the tmr_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tmr_etrg must be configured as follows:

1. The external trigger prescaler should be kept off: ETRGFDIVCFG[1:0] bits are cleared to 00 in the Timer SMCFG register.
2. The external clock mode 1 must be disabled: EXTCLKMODE1EN bit is cleared to 0 in the Timer SMCFG register is cleared to 0.
3. The external trigger polarity (ETRGP) and the external trigger filter (ETRGFILTCFG) can be configured according to the application's needs.

Figure 18-35 shows the behavior of the ocxref signal when the tmr_etrg input becomes high, for both values of the CHxOCREFCLREN enable bit. In this example, the timer is programmed in PWM mode.

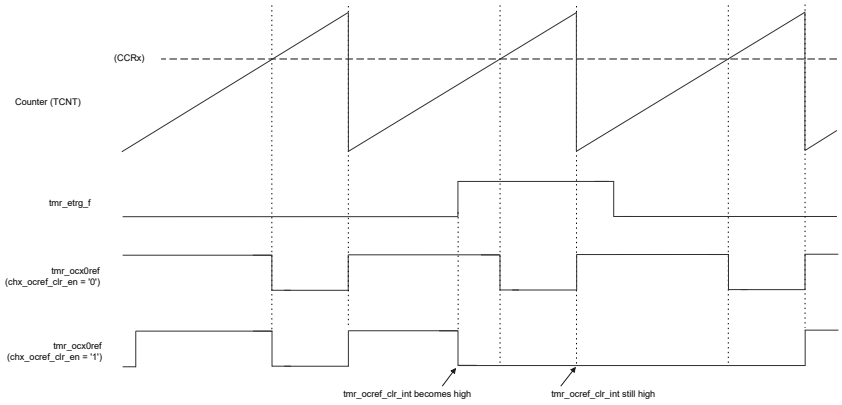


Figure 18-35 Clearing Timer ocxref

NOTE: In case of a PWM with a 100% duty cycle (if CCRx > ARR), ocxref is enabled again at the next counter overflow.

18.3.15 Single-Pulse Mode

The single-pulse mode 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.

The initiation of the counter can be controlled using the slave mode controller. The waveform generation can be performed in either output compare mode or PWM mode. To select the single-pulse mode, the SINGLEPLSMODE bit in the Timer CTRL1 register needs to be set. This causes the counter to automatically stop at the next update event (UEV).

For a pulse to be accurately generated, the compare value must be different from the initial value of the counter. Before starting (when the timer is awaiting the trigger), the configuration must satisfy the condition: TCNT<CCRx ≤ ARR (specifically, 0 < CCRx).

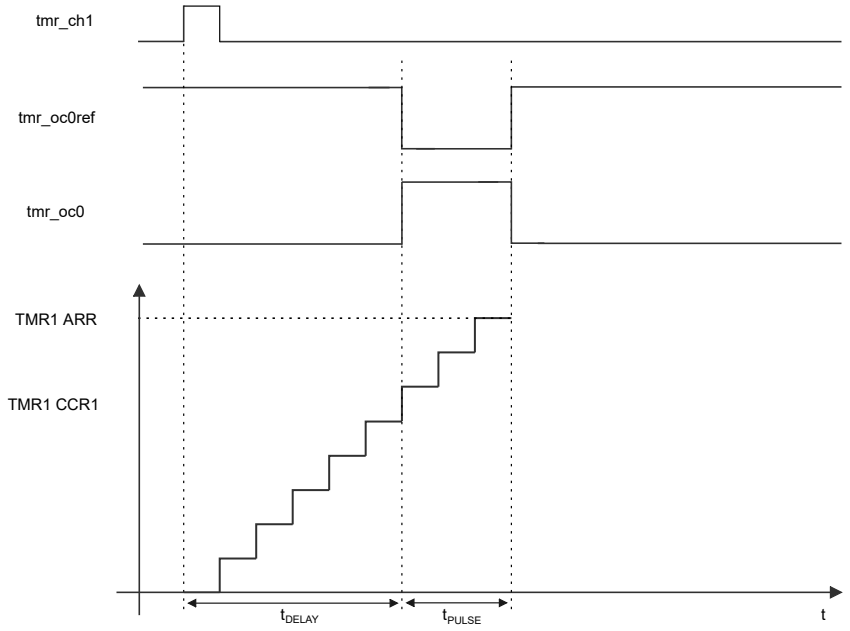


Figure 18-36 Example of Single-Pulse Mode

For example, one may want to generate a positive pulse on tmr_oc0 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tmr_ch1 input pin.

Consider using tmr_ch1fp1 as trigger 1. Here are the steps:

1. Select the proper tmr_ch1_in[7:0] source (internal or external) with the CH1SRCSEL [2:0] bits in the Timer CHxSEL register.
2. Map tmr_ch1fp1 on tmr_ch1 by writing CH1MODESEL = 01 in the Timer CHXMODECTRL1 register.
3. tmr_ch1fp1 must detect a rising edge, write CH1P = 0 and CH1NP = 0 in the Timer CCER register.
4. Configure tmr_ch1fp1 as a trigger for the slave mode controller (tmr_etrg) by writing ITRGSEL = 00110 in the Timer SMCFG register.
5. tmr_ch1fp1 is used to start the counter by writing SLVMODECTRL to 0110 in the Timer SMCFG register (trigger mode).

The SINGLEPLSMODE waveform can be defined by configuring the compare registers, taking into account the clock frequency and the counter prescaler.

- The t_{DELAY} is defined by the value written in the Timer CCR0 register.

- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (Timer ARR - Timer CCR0).
- For example, if the goal is to create 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, PWM mode 2 should be enabled. This can be done by setting CH0OCMODE = 111 in the Timer CHXMODECTRL1 register.

  Optionally, the preload registers can be enabled by setting CH0CCRSHDWEN = 1 in the Timer CHXMODECTRL1 register and ARRSHDWEN in the Timer CTRL1 register.

  In this case, the compare value should be written in the Timer CCR0 register, the auto-reload value in the Timer ARR register, and an update should be generated by setting the SWUPDGEN bit. Finally, an external trigger event on tmr_ch1 should be awaited. In the given example, CH0P is set to 0.

In this example, it is necessary for the DIR and TCNTALIGNMODE bits in the Timer CTRL1 register to be set to a low value. This is because only single-pulse is required. To achieve this, the SINGLEPLSMODE bit in the Timer CTRL1 register should be set to 1 to stop the counter at the next update event, which occurs when the counter reaches the auto-reload value and resets back to 0. On the other hand, if the SINGLEPLSMODE bit is set to 0, the repetitive mode is chosen.

In single-pulse mode, the TCNTEN bit is set by the edge detection on the tmr_chx input, which enables the counter. Subsequently, the comparison between the counter and the compare value makes the output toggle. However, these operations require multiple clock cycles, which limits the minimum delay, t_{DELAY} min, that can be achieved. To generate a waveform with the minimum delay, the CHxOCFASTEN bit can be set in the Timer CCMRx register. This causes the ocxref (and tmr_ocx) to respond to the stimulus without considering the comparison. The new level of ocxref is the same as if a compare match had occurred. The CHxOCFASTEN only functions when the channel is configured in PWM1 or PWM2 mode.

18.3.16 Retriggerable Single-Pulse Mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable single-pulse mode described in Single-Pulse Mode:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in slave mode, with the bits SLVMODECTRL[3:0] = ‘1000’ (combined reset + trigger mode) in the Timer SMCFG register, and the CHxOCMODE[3:0] bits set to ‘1000’ or ‘1001’ for retriggerable single-pulse mode (SINGLEPLSMODE) mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

NOTE: The CHxIF flag does not hold any significance in retriggerable single-pulse mode.

The CHxOCMODE[3:0] and SLVMODECTRL[3:0] bit fields are split into two parts for compatibility reasons; the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have TCNTALIGNMODE[1:0] = 00 in Timer CTRL1.

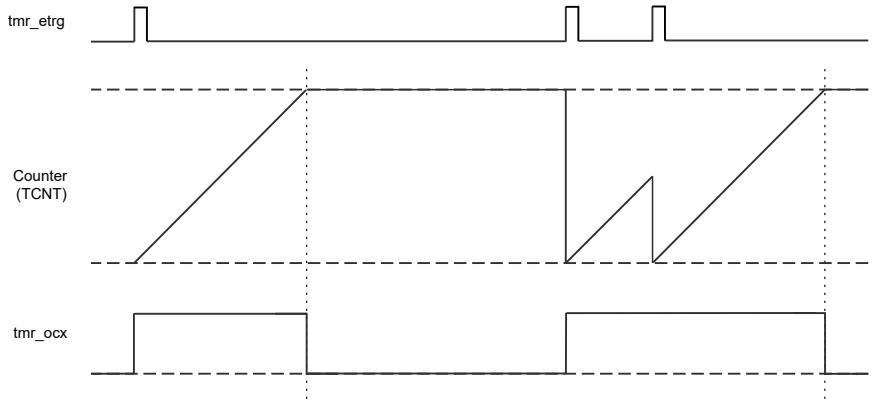


Figure 18-37 Retriggerable Single-Pulse Mode

18.3.17 Encoder Interface Mode

Quadrature Encoder

To select the encoder interface mode, set the SLVMODECTRL value in the Timer SMCFG register.

- If the counter is counting on tmr_ch0 edges only, set SLVMODECTRL to 0001.
- If the counter is counting on tmr_ch1 edges only, set SLVMODECTRL to 0010.
- And if the counter is counting on both tmr_ch0 and tmr_ch1 edges, set SLVMODECTRL to 0011.

To select the polarity of tmr_ch0 and tmr_ch1, program the CH0P and CH1P bits in the Timer CCER register. Ensure that CH0NP and CH1NP remain clear. If necessary, it is also possible to program the input filter.

The two inputs, tmr_ch0 and tmr_ch1, are used to interface to an incremental encoder. For more detailed information, please refer to Table 18-3. The counter is clocked by each valid transition on tmr_ch0fp0 or on tmr_ch1fp1 (tmr_ch0 and tmr_ch1 after input filter and polarity selection, tmr_ch0fp0 = tmr_ch0 if not filtered and not inverted, tmr_ch1fp1 = tmr_ch1 if not filtered and not inverted) assuming that it is enabled (TCNTEN bit in Timer CTRL1 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 Timer CTRL1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tmr_ch0 and tmr_ch1), regardless of whether the counter is counting on tmr_ch0 and/or tmr_ch1.

The 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 Timer ARR register (0 to ARR or ARR down to 0, depending on the direction). So, the Timer ARR must be configured before starting. In the same way, the capture, compare, prescaler, and 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 quadrature encoder and its content; therefore, it always represents the encoder's position.

The count direction corresponds to the rotation direction of the connected sensor. Table 18-3 summarizes the possible combinations, assuming tmr_ch0 and tmr_ch1 do not switch simultaneously.

Table 18-3 Counting Direction versus Encoder Signals (CH0P = CH1P = 0)

Active Edge	SLVMODECTRL[3:0]	Level on Opposite Signal (tmr_ch0fp0 for tmr_ch0, tmr_ch1fp1 for tmr_ch1)	tmr_ch0fp0 Signal		tmr_ch1fp1 Signal	
			Rising	Falling	Rising	Falling
Counting on tmr_ch0 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tmr_ch1 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tmr_ch0 and tmr_ch1 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators normally convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output, which indicates the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 18-38 gives an example of counter operation, showing count signal generation and direction control. It also shows how the input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near one of the switching points. In this example, the configuration is as follows:

- CH0MODESEL = 01 (Timer CHXMODECTRL1 register, tmr_ch0fp0 mapped on tmr_ch0)
- CH1MODESEL = 01 (Timer CHXMODECTRL2 register, tmr_ch1fp1 mapped on tmr_ch1)
- CH0P and CH0NP = 0 (Timer CCER register, tmr_ch0fp0 non-inverted, tmr_ch0fp0 = tmr_ch0)
- CH1P and CH1NP = 0 (Timer CCER register, tmr_ch1fp1 non-inverted, tmr_ch1fp1 = tmr_ch1)
- SLVMODECTRL = 0011 (Timer SMCFG register, both inputs are active on both rising and falling edges)
- TCNTEN = 1 (Timer CTRL1 register, counter is enabled)

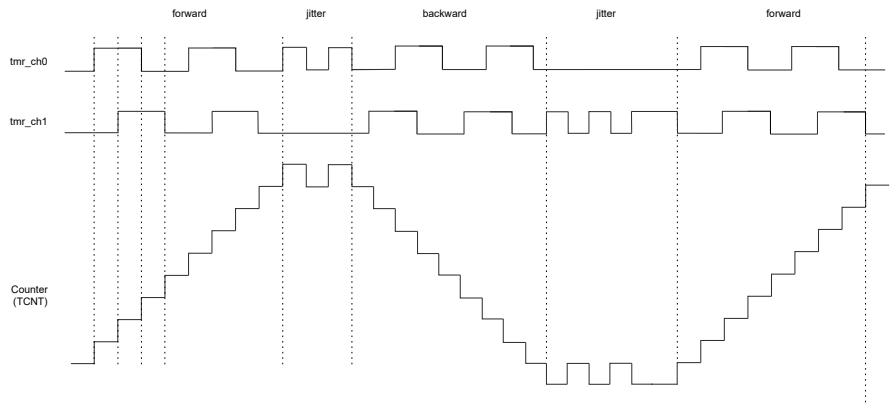


Figure 18-38 Example of Counter Operation in Encoder Interface Mode

Figure 18-39 gives an example of counter behavior when tmr_ch0fp0 polarity is inverted (same configuration as above except CH0P = 1).

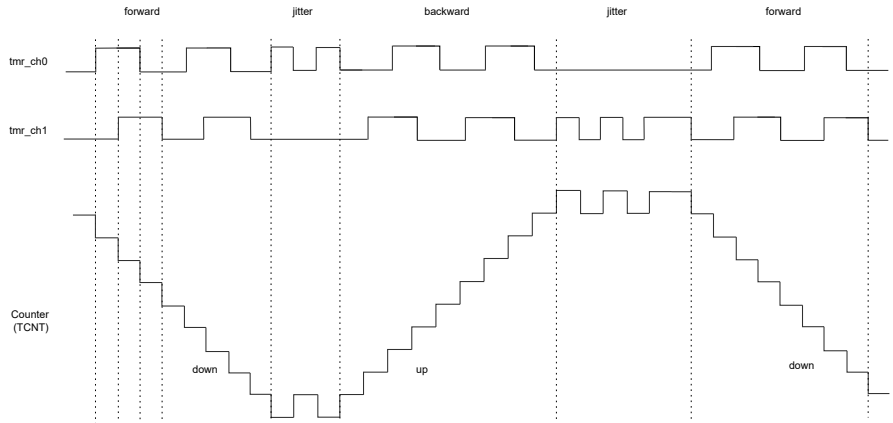


Figure 18-39 Example of Encoder Interface Mode with tmr_ch0fp0 Polarity Inverted

Figure 18-40 shows the timer counter value during a speed reversal for various counting modes.

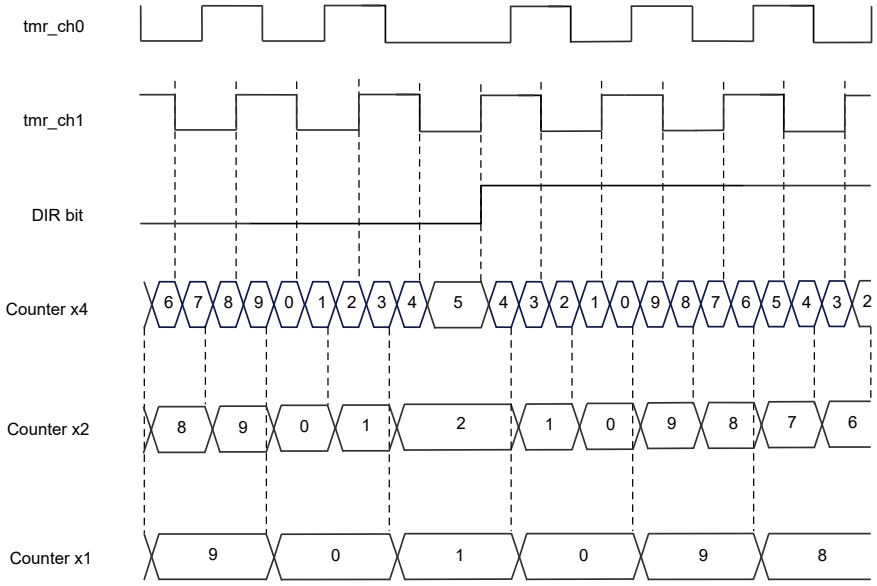


Figure 18-40 Quadrature Encoder Counting Modes

When the timer is configured in encoder interface mode, it provides information on the sensor's current position. Dynamic information can be obtained (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 regularly. 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.

The UPDIFREMAP bit (update interrupt flag status bit remapping) in the Timer CTRL1 register forces a continuous copy of the update interrupt flag (UPDIF) into the bit 31 UPDIFCPY of the timer counter register TCNT. This allows both the counter value and a potential roll-over condition signaled by the UPDIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt). There is no latency between the UPDIF and UPDIFCPY flag assertions.

In 32-bit timer implementations, when the UPDIFREMAP bit is set, bit 31 of the counter register TCNT is overwritten by the UPDIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Encoder Clock Output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tmr_otrg output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MSTMODECTRL[3:0] bit field to 1000 in the Timer CTRL2 register. It is valid for the following SLVMODECTRL[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SLVMODECTRL[3:0] code is not allowed and may lead to unexpected behaviour.

18.3.18 Direction Bit Output

It is possible to output a direction signal out of the timer on the output signals (copy of the DIR bit in the Timer CTRL1 register). This is achieved by setting the CHxOCMODE[3:0] bit field to 1011 in the Timer CHXMODECTRLx register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

18.3.19 UPDIF Bit Remapping

The UPDIFREMAP bit in the Timer CTRL1 register forces a continuous copy of the update interrupt flag (UPDIF) into bit 31 of the timer counter register's bit 31 (TCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UPDIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UPDIF and UPDIFCPY flag assertions.

In 32-bit timer implementations, when the UPDIFREMAP bit is set, bit 31 of the counter is overwritten by the UPDIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

18.3.20 Timer Input XOR Function

The CH0SEL bit in the Timer CTRL2 register, allows the input filter of channel 0 to be connected to the output of a XOR gate, combining the three input pins tmr_ch0, tmr_ch1 and tmr_ch2.

The XOR output can be used with all the timer input functions such as trigger or input capture.

18.3.21 Timers and External Trigger Synchronization

The timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + Trigger mode, and Gated + Reset 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 UPDREQSRC bit from the Timer CTRL1 register is low, an update event (UEV) is generated. Then, all the preloaded registers (ARR, CCRx) are updated.

In the following example, the up-counter is cleared in response to a rising edge on tmr_ch0 input:

1. Configure channel 0 to detect rising edges on tmr_ch0. Configure the input filter duration (in this example, keep CH0ICFLT = 0000 as no filter is required). The capture prescaler is not necessary for triggering, so no configuration is needed. The CH0MODESEL bits select the input capture source only, CH0MODESEL = 01, in the Timer CHXMODECTRL1 register. Write CH0P =0 and CH0NP =0 in the Timer CCER register to validate the polarity and detect rising edges only.
2. Configure the timer in reset mode by writing SLVMODECTRL = 0100 in the Timer SMCFG register. Select tmr_ch0 as the input source by writing ITRGSEL = 00101 in the Timer SMCFG register.
3. Start the counter by writing TCNTEN = 1 in the Timer CTRL1 register.

The counter starts counting on the internal clock, then behaves normally until tmr_ch0 rising edge. When tmr_ch0 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TRGIF bit in the Timer INTSTS register), and an interrupt request or a DMA request can be sent if enabled (depending on the TRGIE and TRGDE bits in Timer DMAINTEN register).

Figure 18-41 shows this behavior when the auto-reload register ARR = 0x36. The delay between the rising edge on tmr_ch0 and the actual reset of the counter is due to the resynchronization circuit on tmr_ch0 input.

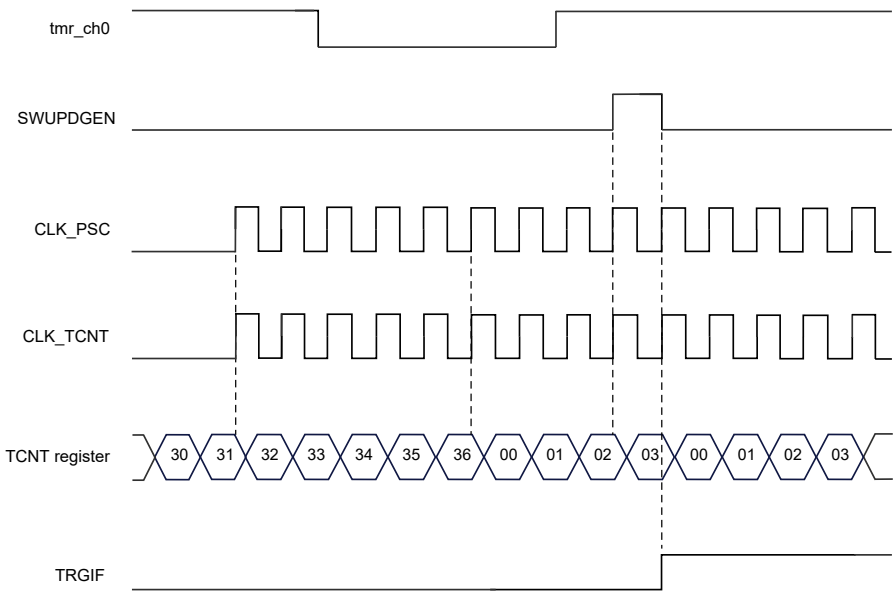


Figure 18-41 Control Circuit in Reset Mode

Slave Mode: Gated Mode

The counter can be enabled depending on the level of a selected input.

In the following example, the up-counter counts only when tmr_ch0 input is low:

1. Configure channel 0 to detect a low level on tmr_ch0. Configure the input filter duration (in this example, keep CH0ICFLT = 0000 as no filter is required). The capture prescaler is not necessary for triggering, so no configuration is needed. The CH0MODESEL bits select the input capture source only, CH0MODESEL = 01, in the Timer CHXMODECTRL1 register.

Write CH0P =1 and CH0NP =0 in the Timer CCER register to validate the polarity (and detect low-level only).

2. Configure the timer in gated mode by writing SLVMODECTRL = 0101 in the Timer SMCFG register. Select tmr_ch0 as the input source by writing ITRGSEL = 00101 in the Timer SMCFG register.
3. Enable the counter by writing TCNTEN = 1 in the Timer CTRL1 register (in gated mode, the counter doesn't start if TCNTEN = 0, whatever the trigger input level).

The counter starts counting on the internal clock as long as tmr_ch0 is low and stops as soon as tmr_ch0 becomes high. The TRGIF flag in the INTSTS register is set when the counter starts or stops.

The delay between the rising edge on tmr_ch0 and the actual stop of the counter is due to the resynchronization circuit on tmr_ch0 input.

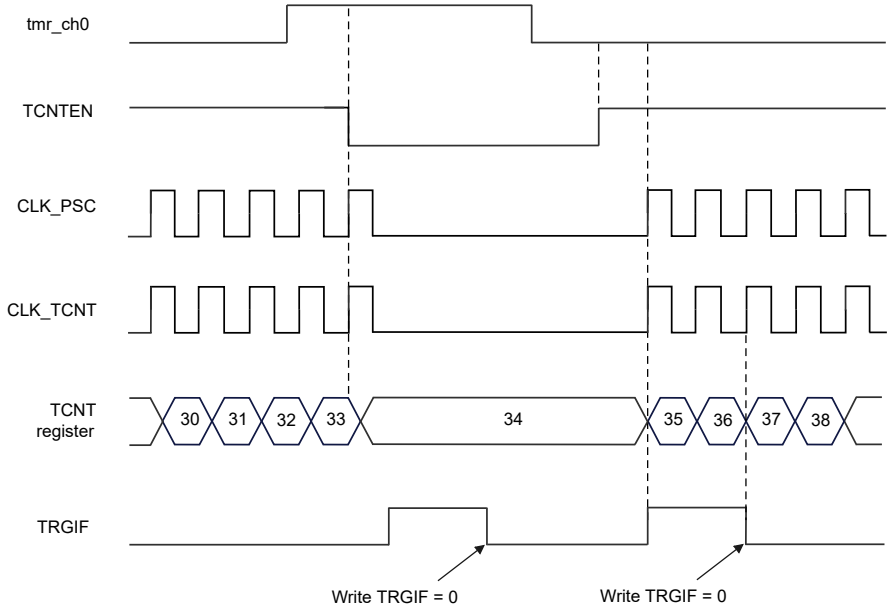


Figure 18-42 Control Circuit in Gated Mode

NOTE: The configuration "CHxP = CHxNP =1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave Mode: Trigger Mode

The counter can start in response to an event on a selected input.

In the following example, the up-counter starts in response to a rising edge on tmr_ch1 input:

1. Configure channel 1 to detect rising edges on tmr_ch1. Configure the input filter duration (in this example, keep CH1ICFLT =0000 as no filter is required). The capture prescaler is not necessary for triggering, so no configuration is needed. CH1MODESEL bits select the input capture source only, CH1MODESEL = 01 in Timer CHXMODECTRL1 register.

   Write CH1P =1 and CH1NP =0 in the Timer CCER register to validate the polarity (and detect low-level only).
2. Configure the timer in trigger mode by writing SLVMODECTRL = 0110 in the Timer SMCFG register. Select tmr_ch1 as the input source by writing ITRGSEL = 00110 in the Timer SMCFG register.

When a rising edge occurs on tmr_ch1, the counter starts counting on the internal clock and the TRGIF flag is set.

The delay between the rising edge on tmr_ch1 and the actual start of the counter is due to the resynchronization circuit on tmr_ch1 input.

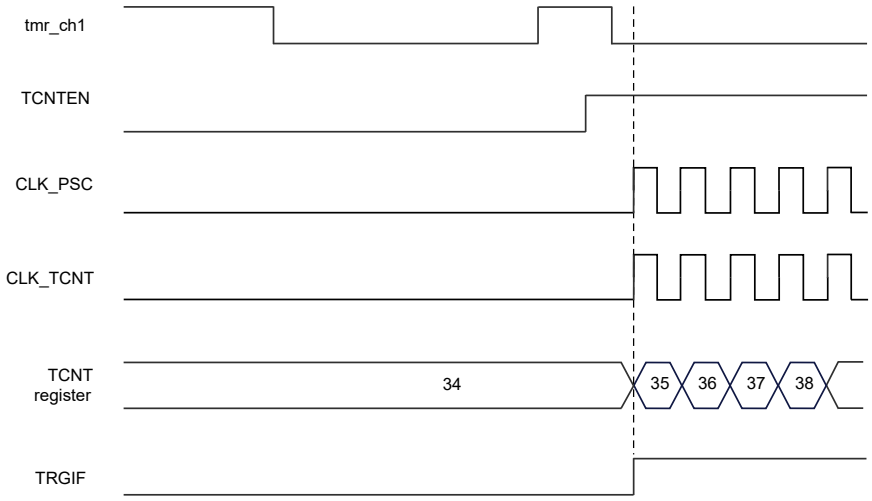


Figure 18-43 Control Circuit in Trigger Mode

Slave Mode – Combined Reset + Trigger Mode

In this case, a rising edge of the selected trigger input (tmr_etrg) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for single-pulse mode.

Slave Mode – Combined Gated + Reset Mode

The counter clock is enabled when the trigger input (tmr_etrg) is high. The counter stops and is reset as soon as the trigger becomes low. Both the start and stop of the counter are controlled.

This mode allows the detection of out-of-range PWM signal (duty cycle exceeding a maximum expected value).

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 tmr_etrg signal is used as the 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 tmr_etrg as tmr_etrg through the ITRGSEL bits of Timer SMCFG register.

In the following example, the up-counter is incremented at each rising edge of the tmr_etrg signal as soon as a rising edge of tmr_ch0 occurs:

1. Configure the external trigger input circuit by programming the Timer SMCFG register as follows:
   - ETRGFILTCFG = 0000: No filter
   - ETRGFDIVCFG =00: Prescaler disabled
   - ETRGP =0: Detection of rising edges on tmr_etrg and EXTCLKMODE1EN = 1 to enable the external clock mode 2.
2. Configure channel 0 as follows to detect rising edges on Timer:
   - CH0ICFLT = 0000: No filter.

- The capture prescaler is not used for triggering and does not need to be configured.
- CH0MODESEL = 01 in the Timer CHXMODECTRL1 register to select only the input capture source
- CH0P =0 and CH0NP =0 in the Timer CCER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SLVMODECTRL = 0110 in the Timer SMCFG register. Select tmr_ch0 as the input source by writing ITRGSEL = 00101 in the Timer SMCFG register.

A rising edge on tmr_ch0 enables the counter and sets the TRGIF flag. The counter then counts on tmr_etrg rising edges.

The delay between the rising edge of the tmr_etrg signal and the actual reset of the counter is due to the resynchronization circuit on the tmr_etrg input.

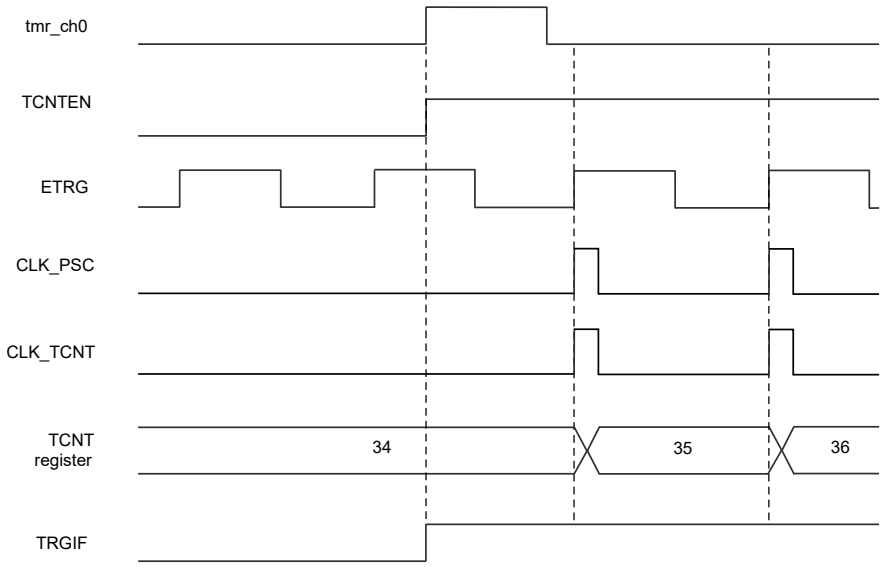


Figure 18-44 Control Circuit in External Clock Mode 2 + Trigger Mode

18.3.22 Timer Synchronization

The 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 18-45 and Figure 18-46 show examples of master/slave timer connections.

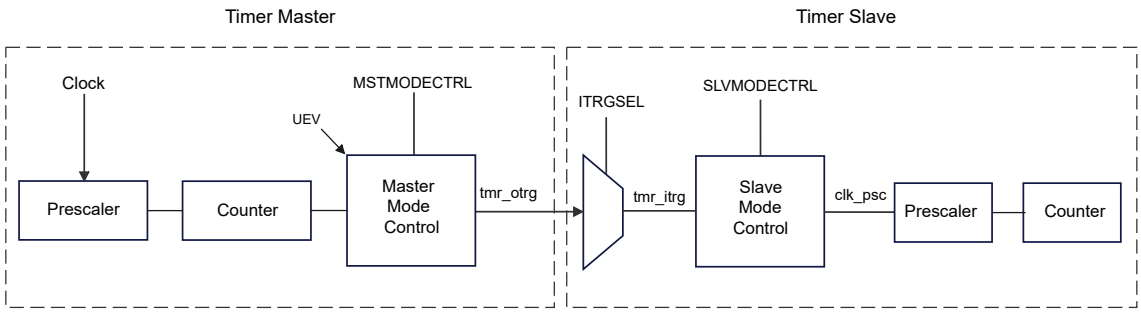


Figure 18-45 Master/Slave Timer Example

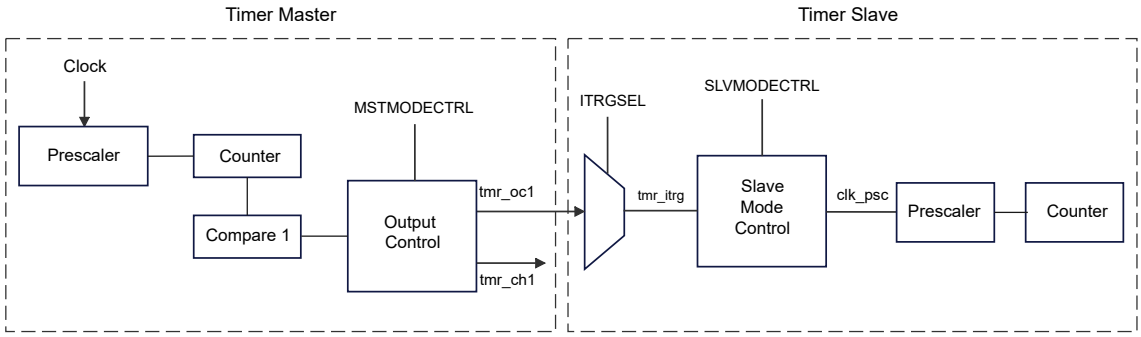


Figure 18-46 Master/Slave Connection Example with 1 Channel Only Timers

NOTE: The timers with one channel only (see Figure 18-46) do not feature a master mode. However, the tmr_oc1 output signal can serve as trigger for slave timer (see Timer internal trigger connection table in Pins and Internal Signals).

The tmr_oc1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination timer CLK_TMR clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

Using one timer as prescaler for another timer

For example, Timer master can be configured to act as a prescaler for Timer slave. Refer to Figure 18-45. To do this:

1. Configure Timer master in master mode so that it outputs a periodic trigger signal on each update event UEV. If MSTMODECTRL = 0010 is written in the Timer CTRL2 register, a rising edge is output on tmr_otrg each time an update event is generated.
2. To connect the tmr_otrg output of Timer master to Timer slave, Timer slave must be configured in slave mode using Internal Trigger 2 (tmr_itr2) as internal trigger. This is selected through the ITRGSEL bits in the Timer SMCFG register (writing ITRGSEL = 00010).
3. Then the slave mode controller must be put in external clock mode 0 (write SLVMODECTRL = 0111 in the TIM SMCFG register). This causes Timer slave to be clocked by the rising edge of the periodic Timer master trigger signal (which correspond to the Timer master counter overflow).
4. Finally both timers must be enabled by setting their respective TCNTEN bits (Timer CTRL1 register).

NOTE: If tmr_ocx is selected on Timer master as the trigger output (MSTMODECTRL = 01xx), its rising edge is used to clock the counter of Timer slave.

Using one timer to enable another timer

In this example, we control the enable of Timer slave with the output compare 1 of Timer master.

Refer to Figure 18-45 for connections. Timer slave counts on the divided internal clock only when tmr_oc1ref of Timer master is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CLK_TMR ($f_{clk\_TCNT} = f_{CLK\_TMR}/3$).

1. Configure Timer master mode to send its Output Compare 1 Reference (tmr_oc0ref) signal as trigger output (MSTMODECTRL = 0100 in the Timer master CTRL2 register).
2. Configure the Timer master tmr_oc0ref waveform (Timer master CHXMODECTRL1 register).

3. Configure Timer slave to get the input trigger from Timer master (ITRGSEL = 00010 in the Timer slave SMCFG register).
4. Configure Timer slave in gated mode (SLVMODECTRL = 0101 in Timer slave SMCFG register).
5. Enable Timer slave by writing ‘1 in the TCNTEN bit (Timer slave CTRL1 register).
6. Start Timer master by writing ‘1 in the TCNTEN bit (Timer master CTRL1 register).

NOTE: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the Timer slave counter enable signal.

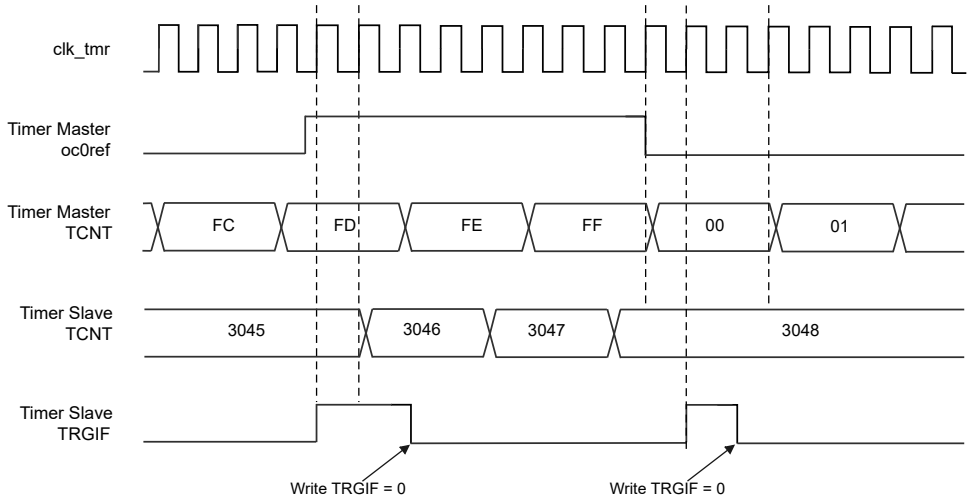


Figure 18-47 Gating Timer Slave with ch0_ocref of Timer Master

In the example in Figure 18-47, the Timer slave 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 master. Then any value can be written in the timer counters. The timers can easily be reset by software using the SWUPDGEN bit in the Timer SWEVTGEN registers.

In the next example (refer to Figure 18-48), we synchronize Timer master and Timer slave. Timer master is the master and starts from 0. Timer slave is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer slave stops when Timer master is disabled by writing ‘0 to the TCNTEN bit in the Timer master CTRL1 register:

1. Configure Timer master mode to send its Output Compare 1 Reference (ch0_ocref) signal as trigger output (MSTMODECTRL = 0100 in the Timer master CTRL2 register).
2. Configure the Timer master ch0_ocref waveform (Timer master CHXMODECTRL1 register).
3. Configure Timer slave to get the input trigger from Timer master (ITRGSEL = 00010 in the Timer slave SMCFG register).
4. Configure Timer slave in gated mode (SLVMODECTRL = 0101 in Timer slave SMCFG register).
5. Reset Timer master by writing 1 in SWUPDGEN bit (Timer master SWEVTGEN register).
6. Reset Timer slave by writing 1 in SWUPDGEN bit (Timer slave SWEVTGEN register).
7. Initialize Timer slave to 0xE7 by writing ‘0xE7’ in the Timer slave counter (TCNT).
8. Enable Timer slave by writing 1 in the TCNTEN bit (Timer slave CTRL1 register).
9. Start Timer master by writing 1 in the TCNTEN bit (Timer master CTRL1 register).
10. Stop Timer master by writing 0 in the TCNTEN bit (Timer master CTRL1 register).

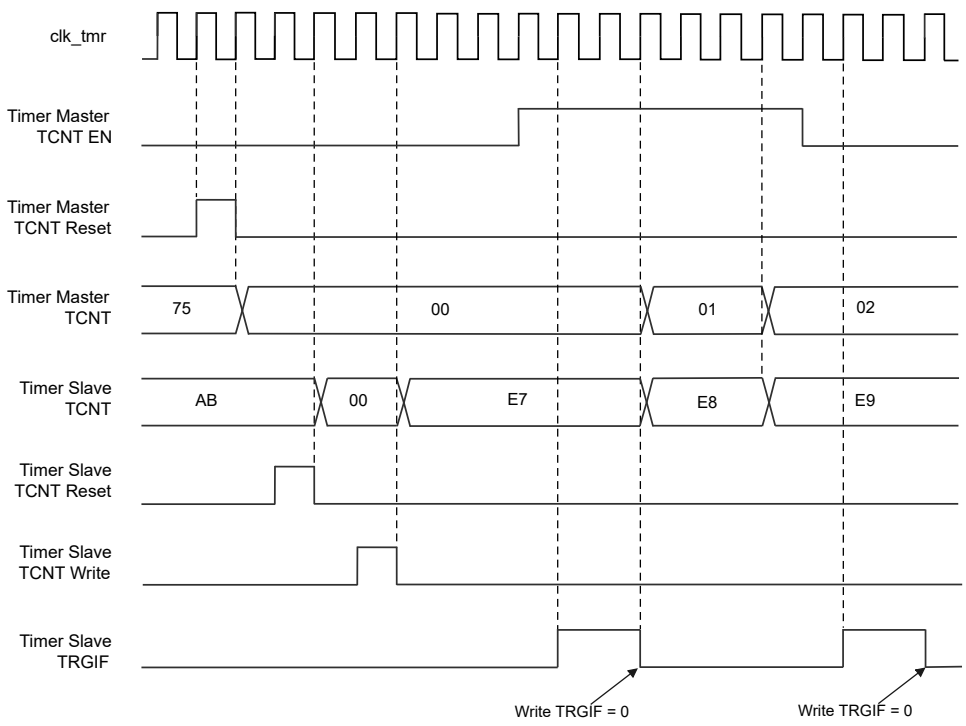


Figure 18-48 Gating Timer Slave with Enable of Timer Master

Using one timer to start another timer

In this example, we set the enable of Timer slave with the update event of Timer master. Refer to Figure 18-45 for connections. Timer slave starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer master. When Timer slave receives the trigger signal its TCNTEN bit is automatically set and the counter counts until we write 0 to the TCNTEN bit in the Timer slave CTRL1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CLK_TMR ($F_{CLK\_TCNT} = f_{CLK\_TMR}/3$).

1. Configure Timer master mode to send its update event (UEV) as trigger output (MSTMODECTRL = 0010 in the Timer master CTRL2 register).
2. Configure the Timer master period (Timer master ARR registers).
3. Configure Timer slave to get the input trigger from Timer master (ITRGSEL = 00010 in the Timer slave SMCFG register).
4. Configure Timer slave in trigger mode (SLVMODECTRL = 0110 in Timer slave SMCFG register).
5. Start Timer master by writing 1 in the TCNTEN bit (Timer master CTRL1 register).

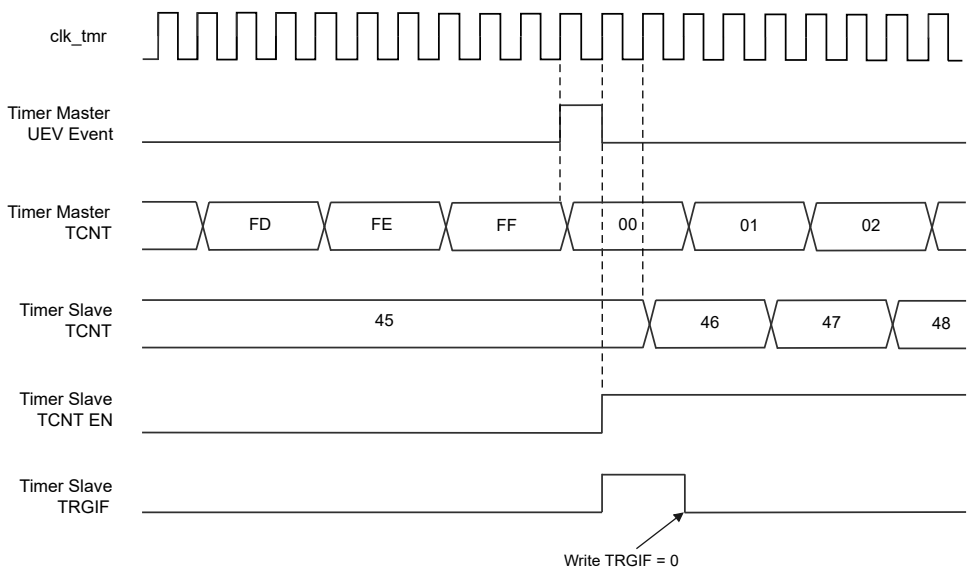


Figure 18-49 Triggering Timer Slave with Update of Timer Master

As in the previous example, both counters can be initialized before starting counting. Figure 18-50 shows the behavior with the same configuration as in Figure 18-49 but in trigger mode instead of gated mode (SLVMODECTRL = 0110 in the Timer slave SMCFG register).

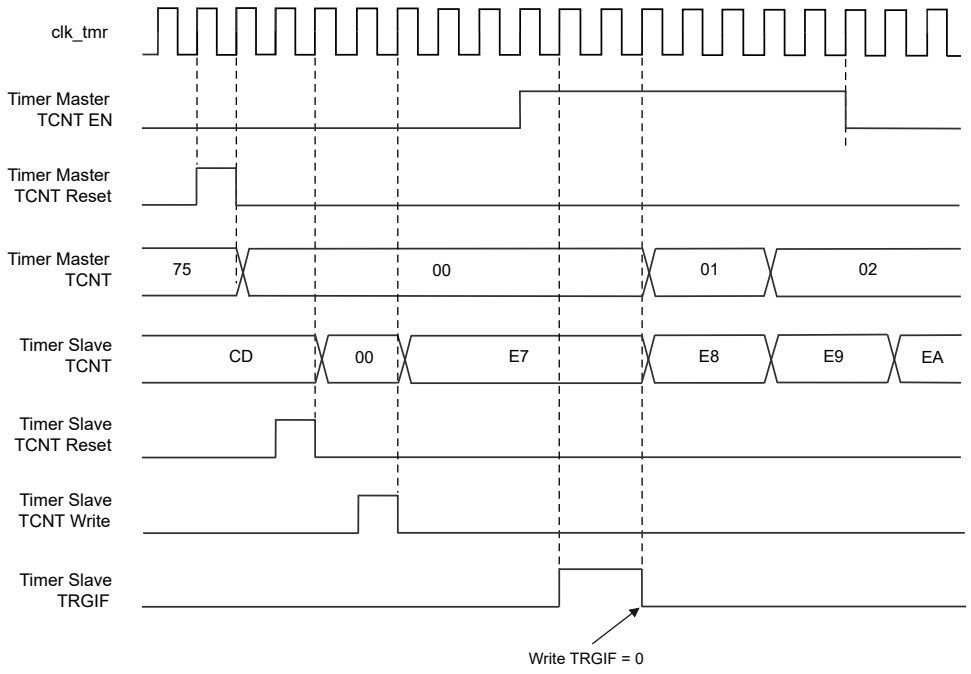


Figure 18-50 Triggering Timer Slave with Enable of Timer Master

Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of Timer master when its tmr_ch0 input rises, and the enable of Timer slave with the enable of Timer master. Refer to Figure 18-45 for connections. To ensure the counters are aligned, Timer master must be configured in Master/Slave mode (slave with respect to tmr_ch0, master with respect to Timer slave):

1. Configure Timer master mode to send its Enable as trigger output (MSTMODECTRL = 0001 in the Timer master CTRL2 register).
2. Configure Timer master slave mode to get the input trigger from tmr_ch0 (ITRGSEL = 00100 in the Timer master SMCFG register).
3. Configure Timer master in trigger mode (SLVMODECTRL = 0110 in the Timer master SMCFG register).
4. Configure the Timer master in Master/Slave mode by writing MSM = 1 (Timer master SMCFG register).
5. Configure Timer slave to get the input trigger from Timer master (ITRGSEL = 00000 in the Timer slave SMCFG register).
6. Configure Timer slave in trigger mode (SLVMODECTRL = 0110 in the Timer slave SMCFG register).

When a rising edge occurs on tmr_ch0 (Timer master), both counters start counting synchronously on the internal clock and both TRGIF flags are set.

NOTE: In this example both timers are initialized before starting (by setting their respective SWUPDGEN bits). Both counters start from 0, but an offset can easily be inserted between them by writing any of the counter registers (TCNT). One can see that the master/slave mode insert a delay between TCNTEN and CLK_PSC on Timer master.

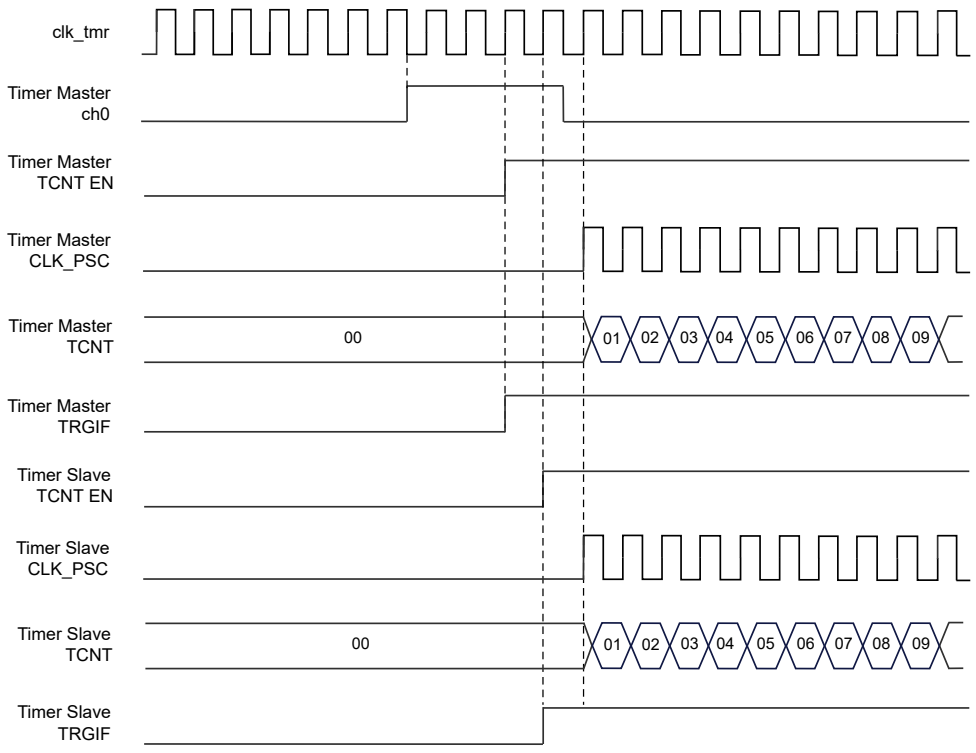


Figure 18-51 Triggering Timer master and Timer slave with Timer master tmr_ch0 input

NOTE: The clock of the slave peripherals (timer, ADC, ...) receiving the tmr_etrg signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

18.3.23 ADC Triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

NOTE: The clock of the slave peripherals (such as timer, ADC) receiving the tmr_etrg signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

18.3.24 DMA Burst Mode

The timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row at regular intervals.

The DMA controller destination is unique and must point to the virtual register Timer DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the Timer DMAR register is redirected to one of the timer registers.

The DMA burst length is determined by the DBL[5:0] bits within the Timer DMACFG register. To obtain more information about these bits, please refer to the description of the DBA bit in the Timer DMACFG register. When a read or write operation occurs at the Timer DMAR address, the timer recognizes it as a burst transfer and counts the number of transfers in half-words or bytes.

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 0, 1, 2, 3) upon an update event, 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 buffer address 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 DMACFG register by configuring the DBA and DBL bit fields: DBL = 3 transfers, DBA = 0xE.
3. Enable the Timer update DMA request (set the UPDDE bit in the Timer DMAINTEN register).
4. Enable Timer.
5. Enable the DMA channel.

This example demonstrates a scenario where each CCRx register needs to be updated once. However, if every CCRx register is to be updated twice, the total number of data to transfer should be 6. For instance, consider a buffer in the RAM that contains data1, data2, data3, data4, data5, and data6. The data is then transferred to the CCRx registers in the following manner:

During the first update DMA request, data1 is transferred to CCR1, data2 is transferred to CCR2, and data3 is transferred to CCR3. On the second update DMA request, data4 is transferred to CCR1, data5 is transferred to CCR2, and data6 is transferred to CCR3.

NOTE: A null value can be written to the reserved registers.

18.3.25 DMA Requests

The GPTMR0/GPTMR1/GPTMR2/GPTMR3 can generate a DMA requests, as shown in Table 18-4.

Table 18-4 Interrupt Requests

DMA Request Signal	DMA Acronym	DMA Request	Enable Control Bit
tmr_updde_dma	TMR_UPDDE	Update	UPDDE
tmr_ch0_dma	TMR_CH0	Capture/compare 0	CH0DE
tmr_ch1_dma	TMR_CH1	Capture/compare 1	CH1DE
tmr_ch2_dma	TMR_CH2	Capture/compare 2	CH2DE
tmr_ch3_dma	TMR_CH3	Capture/compare 3	CH3DE
tmr_etrg_dma	TMR_ETRG	Trigger	TRGDE

NOTE: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section for more details.

18.4 Registers

18.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	GPTMR_CTRL1	TMR control register 1
0x0004	GPTMR_CTRL2	TMR control register 2
0x0008	GPTMR_SMCFG	TMR slave mode control register
0x000c	GPTMR_DMAINTEN	TMR DMA/interrupt enable register
0x0010	GPTMR_INTSTS	TMR status register
0x0014	GPTMR_SWEVTGEN	TMR event generation register
0x0018	GPTMR_CHXMODECTRL1	TMR capture/compare mode register 1
0x001c	GPTMR_CHXMODECTRL2	TMR capture/compare mode register 2
0x0020	GPTMR_CCER	TMR capture/compare enable register
0x0024	GPTMR_TCNT	TMR counter register
0x0028	GPTMR_PSC	TMR prescaler register
0x002c	GPTMR_ARR	TMR auto-reload register
0x0030	GPTMR_RCR	TMR repetition counter register
0x0034	GPTMR_CCR0	TMR capture/compare register 0
0x0038	GPTMR_CCR1	TMR capture/compare register 1
0x003c	GPTMR_CCR2	TMR capture/compare register 2
0x0040	GPTMR_CCR3	TMR capture/compare register 3
0x0044	GPTMR_BDTR	TMR break and dead-time register

Offset	Register Name	Register Description
0x0048	GPTMR_CHXSEL	TMR timer input selection register
0x004c	GPTMR_AF1	TMR alternate function option register 1
0x0050	GPTMR_AF2	TMR alternate function option register 2
0x0054	GPTMR_DMACFG	TMR DMA control register
0x0058	GPTMR_DMAR	TMR DMA address for full transfer

18.4.2 Register Field Details

18.4.2.1 GPTMR_CTRL1

0x0000			TMR control register 1											GPTMR_CTRL1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				UPDIF REMAP	Reserv ed	CLKDTFDIV		ARRSH DWEN	TCNTALIGNMODE		DIR	SINGLE PLSMO DE	UPDRE QSRC	UPDDI S	TCNTE N
Type	RO				RW	RO	RW		RW	RW		RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-5 TMR Control Register 1 Description

Field	Name	Description
31:12	Reserved	Reserved
11	UPDIFREMAP	UPDIF status bit remapping 0: No remapping. UPDIF status bit is not copied to TCNT register bit 31. 1: Remapping enabled. UPDIF status bit is copied to TCNT register bit 31.
10:10	Reserved	Reserved
9:8	CLKDTFDIV	Clock division This bitfield indicates the division ratio between the timer clock (CLK_TMR) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (etrg, chx), 00: $t_{DTS} = t_{CLK\_TMR}$

Field	Name	Description
		01: t_{DTS} = 2 x t_{CLK_TMR} 10: t_{DTS} = 4 x t_{CLK_TMR} 11: Reserved, do not program this value
7	ARRSHDWEN	Auto-reload register shadow enable 0: TMR_ARR register is not buffered 1: TMR_ARR register is buffered
6:5	TCNTALIGNMODE	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 (CHxMODESEL = 00 in CHxMODECTRLx 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 (CHxMODESEL = 00 in CHxMODECTRLx 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 (CHxMODESEL = 00 in CHxMODECTRLx register) are set 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 (TCNTEN = 1).
4	DIR	Direction 0: Counter used as up-counter 1: Counter used as down-counter **NOTE:** This bit is read only when the timer is configured in center-aligned mode or encoder mode.
3	SINGLEPLSMODE	Single pulse mode 0: Counter is not stopped at update event 1: Counter stops counting at the next update event (clearing the bit TCNTEN)

Field	Name	Description
2	UPDREQSRC	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 'SWUPDGEN' bit • Update generation through the slave mode controller 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.
1	UPDDIS	Update disable This bit is set and cleared by software to enable/disable UEV event generation. 0: UEV enabled. The update event (UEV) is generated by one of the following events: • Counter overflow/underflow • Setting the 'SWUPDGEN' 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 'SWUPDGEN' bit is set or a hardware reset is received from the slave mode controller.
0	TCNTEN	Counter enable 0: Counter disabled 1: Counter enabled

Field	Name	Description
		NOTE: External clock, gated mode, and encoder mode can work only if the 'TCNTEN' bit has been previously set by software. However, trigger mode can set the 'TCNTEN' bit automatically by hardware.If SINGLEPLSMODE = 1, TCNTEN will be clear by hardware when UEV happen.

18.4.2.2 GPTMR_CTRL2

0x0004			TMR control register 2											GPTMR_CTRL2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved							MSTMODECTRLBIT3	Reserved							
Type	RO							RW	RO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH3NOIS	CH3OIS	CH2NOIS	CH2OIS	CH1NOIS	CH1OIS	CH0NOIS	CH0OIS	CH0SEL	MSTMODECTRL			CCDMAREQSRC	CCUPDSRC	Reserved	CCSHWDEN
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW			RW	RW	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-6 TMR Control Register 2 Description

Field	Name	Description
31:25	Reserved	Reserved
24	MSTMODECTRLBIT3	The bit 3 of MSTMODECTRL1 Refer to [6:4]
23:16	Reserved	Reserved
15	CH3NOIS	CH3N output idle state
14	CH3OIS	CH3 output idle state Refer to CH0OIS.
13	CH2NOIS	CH2N output idle state Refer to CH0NOIS.

Field	Name	Description
12	CH2OIS	CH2 output idle state Refer to CH0OIS.
11	CH1NOIS	CH1N output idle state Refer to CH0NOIS.
10	CH1OIS	CH1 output idle state Refer to CH0OIS.
9	CH0NOIS	CH0N output idle state 0: tmr_oc1n = 0 after a dead-time when MOE = 0 1: tmr_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 the BDTR register).
8	CH0OIS	CH0 output idle state 0: tmr_oc1 = 0 (after a dead-time) when MOE = 0 1: tmr_oc1=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 the BDTR register).
7	CH0SEL	TI0 selection 0: The tmr_ti0_in[7..0] multiplexer output is connected to tmr_ti0 input 1: tmr_ti0_in[7..0], tmr_ti1_in[7..0] and tmr_ti2_in[7..0] multiplexers outputs are XORed and connected to the tmr_ti0 input
6:4	MSTMODECTRL	Master mode selection These bits select the information to be sent in master mode to slave timers for synchronization (tmr_otrg). The combination is as follows:

Field	Name	Description
		0000: Reset - the 'SWUPDGEN' bit from theSWEVTGEN register is used as trigger output (tmr_otrg). If the reset is generated by the trigger input (slave mode controller configured in reset mode), then the signal on tmr_otrg is delayed compared to the actual reset. 0001: Enable - the Counter Enable signal CNT_EN is used as trigger output (tmr_otrg). It is useful to start several timers simultaneously or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the 'TCNTEN' 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 tmr_otrg, except if the master/slave mode is selected (see the MSM bit description in SMCFG register). 0010: Update - The update event is selected as trigger output (tmr_otrg). For instance, a master timer can then be used as a prescaler for a slave timer. 0011: Compare Pulse - The trigger output sends a positive pulse when the CH0IF flag is set (even if it was already high) as soon as a capture or a compare match occurs. (tmr_otrg). 0100: Compare - tmr_oc0refc signal is used as trigger output (tmr_otrg) 0101: Compare - tmr_oc1refc signal is used as trigger output (tmr_otrg) 0110: Compare - tmr_oc2refc signal is used as trigger output (tmr_otrg) 0111: Compare - tmr_oc3refc signal is used as trigger output (tmr_otrg) 1000: Encoder Clock output - The encoder clock signal is used as trigger output (tmr_otrg). This code is valid for the following 'SLVMODECTRL'[3:0] values: 0001, 0010, 0011. Any other 'SLVMODECTRL'[3:0] code is not allowed and may lead to unexpected behavior. Other codes reserved

Field	Name	Description
		NOTE: The clock of the slave timer or 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	CCDMAREQSRC	Capture/compare DMA selection 0: CCx DMA request sent when CCx event occurs 1: CCx DMA requests sent when update event occurs
2	CCUPDSRC	Capture/compare control update selection 0: When capture/compare control bits are preloaded (CCSHWDEN = 1), they are updated by setting the SWCOMGEN bit only. 1: When capture/compare control bits are preloaded (CCSHWDEN = 1), they are updated by setting the SWCOMGEN bit or when a rising edge occurs on tmr_itrg. **NOTE:** This bit acts only on channels that have a complementary output.
1	Reserved	Reserved
0	CCSHWDEN	Capture/compare preloaded control 0: CHxEN, CHxNEN, and CHxOCMODE bits are not preloaded. 1: CHxEN, CHxNEN, and CHxOCMODE bits are preloaded. After being written, they are updated only when a commutation event (COM) occurs (SWCOMGEN bit set or rising edge detected on tmr_itrg, depending on the CCUPDSRC bit). **NOTE:** This bit acts only on channels that have a complementary output.

18.4.2.3 GPTMR_SMCFG

0x0008			TMR slave mode control register											GPTMR_SMCFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										ITRGSELBIT4	ITRGSELBIT3	Reserved			OCREFCLRSEL
Type	RO										RW	RW	RO			RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ETRGP	EXTCLKMODE1EN	ETRGFDIVCFG		ETRGFILTCFG				MSM	ITRGSEL			SLVMODECTRL			
Type	RW	RW	RW		RW				RW	RW			RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-7 TMR Slave Mode Control Register Description

Field	Name	Description
31:22	Reserved	Reserved
21	ITRGSELBIT4	Trigger selection - bit 4 Refer to 'ITRGSEL'[2:0] description - bits 6:4
20	ITRGSELBIT3	Trigger selection - bit 3 Refer to 'ITRGSEL'[2:0] description - bits 6:4
19:17	Reserved	Reserved
16	OCREFCLRSEL	OCREF clear selection This bit is used to select the OCREF clear source. 0: tmr_ocref_clr_int is connected to the tmr_ocref_clr input

Field	Name	Description
		1: tmr_ocref_clr_int is connected to tmr_etrf
15	ETRGP	External trigger polarity This bit selects whether tmr_etr_in or tmr_etr_in invert is used for trigger operations 0: tmr_etr_in is non-inverted, active at high level or rising edge. 1: tmr_etr_in is inverted, active at low level or falling edge.
14	EXTCLKMODE1EN	External clock enable This bit enables external clock mode 1. 0: External clock mode 1 disabled 1: External clock mode 1 enabled. The counter is clocked by any active edge on the tmr_etrf signal. **NOTE:** Setting the 'EXTCLKMODE1EN' bit has the same effect as selecting external clock mode 0 with tmr_itrg connected to tmr_etrf ('SLVMODECTRL'=111 andITRGSEL = 00111). It is possible to simultaneously use external clock mode 1 with the following slave modes: reset mode, gated mode, and trigger mode. Nevertheless, tmr_itrg must not be connected to tmr_etrf in this case ('ITRGSEL' bits must not be 00111). If external clock mode 0 and external clock mode 1 are enabled simultaneously, the external clock input is tmr_etrf.
13:12	ETRGFDIVCFG	External trigger prescaler External trigger signal tmr_etrp frequency must be at most 1/4 of TMRCLK frequency. A prescaler can be enabled to reduce tmr_etrp frequency. It is useful when inputting fast external clocks on tmr_etr_in. 00: Prescaler OFF 01: tmr_etr_in frequency divided by 2 10: tmr_etr_in frequency divided by 4 11: tmr_etr_in frequency divided by 8

Field	Name	Description
11:8	ETRGFILTCFG	External trigger filter This bitfield then defines the frequency used to sample the tmr_etrp signal and the length of the digital filter applied to tmr_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: $f_{SAMPLING}$ = fclk_tmr, N = 2 0010: $f_{SAMPLING}$ = fclk_tmr, N = 4 0011: $f_{SAMPLING}$ = fclk_tmr, 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	MSM	Master/slave mode 0: No action 1: The effect of an event on the trigger input (tmr_itrg) is delayed to allow a perfect synchronization between the current timer and its slaves (through tmr_otrg). It is useful if we want to synchronize several timers on a single external event.
6:4	ITRGSEL	Trigger selection

Field	Name	Description
		This bitfield is combined with 'ITRGSEL'[4:3] bits. This bitfield selects the trigger input to be used to synchronize the counter. 00000: Internal trigger 0 (tmr_itr0) 00001: Internal trigger 1 (tmr_itr1) 00010: Internal trigger 2 (tmr_itr2) 00011: Internal trigger 3 (tmr_itr3) 00100: tmr_ti1 edge detector (tmr_ti1f_ed) 00101: Filtered timer input 1 (tmr_ch1fp0) 00110: Filtered timer input 2 (tmr_ti2fp2) 00111: External trigger input (tmr_etrf) 01000: Internal trigger 0 (tmr_itr4) 01001: Internal trigger 1 (tmr_itr5) 01010: Internal trigger 1 (tmr_itr6) 01011: Internal trigger 1 (tmr_itr7) 01100: Internal trigger 1 (tmr_itr8) 01101: Internal trigger 1 (tmr_itr9) 01110: Internal trigger 1 (tmr_itr10) 01111: Internal trigger 1 (tmr_itr11) 10000: Internal trigger 1 (tmr_itr12) 10001: Internal trigger 1 (tmr_itr13) 10010: Internal trigger 1 (tmr_itr14) 10011: Internal trigger 1 (tmr_itr15) Others: Reserved **NOTE:** These bits must be changed only when they are not used (e.g. when 'SLVMODECTRL'=0000) to avoid wrong edge detections at the transition.
3:0	SLVMODECTRL	Slave mode selection

Field	Name	Description
		When external signals are selected, the active edge of the trigger signal (tmr_itrg) is linked to the polarity selected on the external input (see input control register and control register description. 0000: Slave mode disabled. If TCNTEN = 1, the prescaler is clocked directly by the internal clock. 0001: Quadrature encoder mode 1, x2 mode. Counter counts up/down on tmr_ch0fp0 edge depending on tmr_ti1fp2 level. 0010: Quadrature encoder mode 2, x2 mode. Counter counts up/down on tmr_ti1fp2 edge depending on tmr_ch0fp0 level. 0011: Quadrature encoder mode 3, x4 mode. Counter counts up/down on both tmr_ch0fp0 and tmr_ti1fp2 edges depending on the level of the other input. 0100: Reset Mode. The rising edge of the selected trigger input (tmr_itrg) reinitializes the counter and generates an update of the registers. 0101: Gated mode. The counter clock is enabled when the trigger input (tmr_itrg) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both the start and stop of the counter are controlled. 0110: Trigger mode - The counter starts at a rising edge of the trigger tmr_itrg (but it is not reset). Only the start of the counter is controlled. 0111: External clock mode 0 - Rising edges of the selected trigger (tmr_itrg) clock the counter. 1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tmr_itrg) reinitializes the counter, generates an update of the registers, and starts the counter. 1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tmr_itrg) is high. The counter stops and is reset) as soon as the trigger becomes low. Both the start and stop of the counter are controlled.

Field	Name	Description
		NOTE: The gated mode must not be used if ch0_fp_ed is selected as the trigger input (ITRGSEL = 00100). Indeed, ch0_fp_ed outputs 1 pulse for each transition on TI0F, whereas the gated mode checks the level of the trigger signal. The clock of the slave peripherals (timer, ADC, ...) receiving the tmr_otrg or the tmr_otrg2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

18.4.2.4 GPTMR_DMAINTEN

0x000c			TMR DMA/interrupt enable register											ADVTMR_DMAINTEN		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										DIRIE	Reserved				
Type	RO										RW	RO				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	TRGDE	COMDE	CH3DE	CH2DE	CH1DE	CH0DE	UPDDE	BRKIE	TRGIE	COMIE	CH3IE	CH2IE	CH1IE	CH0IE	UPDIE
Type	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-8 TMR DMA/Interrupt Enable Register Description

Field	Name	Description
31:22	Reserved	Reserved
21	DIRIE	Direction change interrupt enable 0: Direction change interrupt disabled 1: Direction change interrupt enabled
20:15	Reserved	Reserved
14	TRGDE	Trigger DMA request enable 0: Trigger DMA request disabled 1: Trigger DMA request enabled
13	COMDE	COM DMA request enable 0: COM DMA request disabled 1: COM DMA request enabled

Field	Name	Description
12	CH3DE	Capture/compare 3 DMA request enable 0: CC3 DMA request disabled 1: CC3 DMA request enabled
11	CH2DE	Capture/compare 2 DMA request enable 0: CC2 DMA request disabled 1: CC2 DMA request enabled
10	CH1DE	Capture/compare 1 DMA request enable 0: CC1 DMA request disabled 1: CC1 DMA request enabled
9	CH0DE	Capture/compare 0 DMA request enable 0: CC0 DMA request disabled 1: CC0 DMA request enabled
8	UPDDE	Update DMA request enable 0: Update DMA request disabled 1: Update DMA request enabled
7	BRKIE	Break interrupt enable 0: Break interrupt disabled 1: Break interrupt enabled
6	TRGIE	Trigger interrupt enable 0: Trigger interrupt disabled 1: Trigger interrupt enabled
5	COMIE	COM interrupt enable 0: COM interrupt disabled 1: COM interrupt enabled
4	CH3IE	Capture/compare 3 interrupt enable

Field	Name	Description
		0: CC3 interrupt disabled 1: CC3 interrupt enabled
3	CH2IE	Capture/compare 2 interrupt enable 0: CC2 interrupt disabled 1: CC2 interrupt enabled
2	CH1IE	Capture/compare 1 interrupt enable 0: CC1 interrupt disabled 1: CC1 interrupt enabled
1	CH0IE	Capture/compare 0 interrupt enable 0: CC0 interrupt disabled 1: CC0 interrupt enabled
0	UPDIE	Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

18.4.2.5 GPTMR_INTSTS

0x0010			TMR status register											GPTMR_INTSTS		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										DIRF	Reserved				
Type	RO										RC_W0	RO				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		SYSBRKIF	CH3OF	CH2OF	CH1OF	CH0OF	Reserved	BRKIF	TRGIF	COMIF	CH3IF	CH2IF	CH1IF	CH0IF	UPDIF
Type	RO		RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RO	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-9 TMR Status Register Description

Field	Name	Description
31:22	Reserved	Reserved
21	DIRF	Direction change interrupt flag This flag is set by hardware when the direction changes in encoder mode. It is cleared by software by writing it to '0'. 0: No direction change 1: Direction change
20:14	Reserved	Reserved
13	SYSBRKIF	System break interrupt flag This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active. This flag must be reset to restart the PWM operation. 0: No break event occurred.

Field	Name	Description
		1: An active level has been detected on the system break input. An interrupt is generated if BRKIE = 1 in the Timer DMAINTEN register.
12	CH3OF	Capture/compare 3 overcapture flag Refer to CH0OF description
11	CH2OF	Capture/compare 2 overcapture flag Refer to CH0OF description
10	CH1OF	Capture/compare 1 overcapture flag Refer to CH0OF description
9	CH0OF	Capture/compare 0 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 CCR0 register while the CH0IF flag was already set
8:8	Reserved	Reserved
7	BRKIF	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. An interrupt is generated if BRKIE = 1 in the Timer DMAINTEN register.
6	TRGIF	Trigger interrupt flag This flag is set by hardware on the TRG trigger event (active edge detected on tmr_itrg input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

Field	Name	Description
		0: No trigger event occurred. 1: Trigger interrupt pending.
5	COMIF	COM interrupt flag This flag is set by hardware on COM event (when capture/compare control bits - CHxEN, CHxNEN, CHxOCMODE - have been updated). It is cleared by software. 0: No COM event occurred 1: COM interrupt pending
4	CH3IF	Capture/compare 3 interrupt flag Refer to CH0IF description
3	CH2IF	Capture/compare 2 interrupt flag Refer to CH0IF description
2	CH1IF	Capture/compare 1 interrupt flag Refer to CH0IF description
1	CH0IF	Capture/compare 0 interrupt flag This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the CCR0 register (input capture mode only). 0: No compare match or input capture occurred 1: A compare match or an input capture occurred If channel CH0 is configured as output: This flag is set when the content of the counter TCNT matches the content of the CCR0 register. When the content of CCR0 is greater than ARR's, the CH0IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode. Refer to the 'TCNTALIGNMODE' bits in the CTRL1 register for the full description. If channel CH0 is configured as input:

Field	Name	Description
		This bit is set when the counter value has been captured in CCR0 register (an edge has been detected on IC0, as per the edge sensitivity defined with the 'CH0P' and 'CH0NP' bits setting, in CCER).
0	UPDIF	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 UPDDIS = 0 in the CTRL1 register. • When CNT is reinitialized by software using the 'SWUPDGEN' bit in SWEVTGEN register, if UPDREQSRC = 0 and UPDDIS = 0 in the CTRL1 register. • When CNT is reinitialized by a trigger event , if UPDREQSRC = 0 and UPDDIS = 0 in the CTRL1 register.

18.4.2.6 GPTMR_SWEVTGEN

0x0014			TMR event generation register											GPTMR_SWEVTGEN		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								SWBRKGEN	SWTRGGEN	SWCOMGEN	SWCH3CCGEN	SWCH2CCGEN	SWCH1CCGEN	SWCH0CCGEN	SWUPDGEN
Type	RO								WC	WC	WC	WC	WC	WC	WC	WC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-10 TMR Event Generation Register Description

Field	Name	Description
31:8	Reserved	Reserved
7	SWBRKGEN	Break generation This bit is set by software to generate an event, it is automatically cleared by hardware. 0: No action 1: A break event is generated. MOE bit is cleared and BRKIF flag is set. Related interrupt or DMA transfer can occur if enabled.
6	SWTRGGEN	Trigger generation This bit is set by software to generate an event, it is automatically cleared by hardware. 0: No action 1: The TRGIF flag is set in INTSTS register. Related interrupt or DMA transfer can occur if enabled.
5	SWCOMGEN	Capture/compare control update generation

Field	Name	Description
		This bit can be set by software, it is automatically cleared by the hardware. 0: No action 1: When the 'CCSHWDEN' bit is set, it allows to update CHxEN, CHxNEN and CHxOCMODE bits. **NOTE:** This bit acts only on channels having a complementary output.
4	SWCH3CCGEN	Capture/compare 3 generation Refer to SWCH0CCGEN description
3	SWCH2CCGEN	Capture/compare 2 generation Refer to SWCH0CCGEN description
2	SWCH1CCGEN	Capture/compare 1 generation Refer to SWCH0CCGEN description
1	SWCH0CCGEN	Capture/compare 0 generation This bit is set by software 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: CH0IF 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 CCR0 register. The CH0IF flag is set, and the corresponding interrupt or DMA request is sent if enabled. The CH0OF flag is set if the CH0IF flag was already high.
0	SWUPDGEN	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.

Field	Name	Description
		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 (up-counting), else it takes the auto-reload value (ARR) if DIR = 1 (down-counting).

18.4.2.7 GPTMR_CHXMODECTRL1

0x0018	TMR capture/compare mode register 1													GPTMR_CHXMODECTRL1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved							CCMR1								
Type	RO							RW								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CCMR1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-11 TMR capture/compare Mode Register 1 Description

Field	Name	Description
31:25	Reserved	Reserved
24:0	CCMR1	The field of this register will have different functions when the channel is in different modes (input capture/output compare) **Output compare mode**: [24]: CH1OCMODE_bit3; The bit3 of CH1OCMODE [16]: CH0OCMODE_bit3; The bit3 of CH0OCMODE [15]: CH1OCREFCLREN; CH1 Output Compare OCxREF clear enable. Refer to CH0OCREFCLREN. [14:12]: CH1OCMODE; CH1 Output Compare mode selection. Refer to CH0OCMODE. [11]: CH1CCRSHDWEN; CH1 Output Compare shadow enable. Refer to CH0CCRSHDWEN. [10]: CH1OCFASTEN; CH1 Output Compare fast enable. Refer to CH0OCFASTEN. [9:8]: CH1MODESEL; CH1 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH1 channel is configured as output

Field	Name	Description
		01: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_ch1 10: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_ch0 11: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through the 'ITRGSEL' bit (SMCFG register) **NOTE:** CH1MODESEL bits are writable only when the channel is off (CH1EN = '0' in CCER). [7]: CH0OCREFCLREN; CH0 Output Compare OCxREF clear enable 0: tmr_oc1ref is not affected by the tmr_ocref_clr_int signal 1: tmr_oc1ref is cleared as soon as a High level is detected on tmr_ocref_clr_int signal (tmr_ocref_clr input or tmr_etrf input) [6:4]: CH0OCMODE; CH0 Output Compare mode selection These bits define the behavior of the output reference signal tmr_oc0ref from which tmr_oc0 and tmr_oc0n are derived. tmr_oc0ref is active high whereas tmr_oc0 and tmr_oc0n active level depends on 'CH0P' and 'CH0NP' bits. 0000: Frozen - The comparison between the output compare register CCR0 and the counter TCNT has no effect on the outputs. (This mode is used to generate a timing base). 0001: Set channel 0 to active level on match. tmr_oc0ref signal is forced high when the counter TCNT matches the capture/compare register 0 (CCR0). 0010: Set channel 0 to inactive level on match. tmr_oc0ref signal is forced low when the counter TCNT matches the capture/compare register 0 (CCR0). 0011: Toggle - tmr_oc0ref toggles when TCNT = CCR0. 0100: Force inactive level - tmr_oc0ref is forced low. 0101: Force active level - tmr_oc0ref is forced high. 0110: PWM mode 1 - In up-counting, channel 0 is active as long as TCNT is less than CCR0 else inactive. In down-counting, channel 0 is inactive (tmr_oc0ref = '0') as long as TCNT is great than CCR0 else active (tmr_oc0ref = '1'). 0111: PWM mode 2 - In up-counting, channel 0 is inactive as long as TCNT less than CCR0 else active. In down-counting, channel 0 is active as long as TCNT is great than CCR0 else inactive. 1000: Retrigerrable SINGLEPLSMODE mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become active again at the next update. In

Field	Name	Description
		down-counting mode, the channel is inactive until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become inactive at the next update. 1001: Retrigerrable SINGLEPLSMODE mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 2 and the channels become inactive at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become active again at the next update. 1010: Reserved 1011: Reserved 1100: Combined PWM mode 1 - tmr_oc0ref has the same behavior as in PWM mode 1. tmr_oc0refc is the logical OR between tmr_oc0ref and tmr_oc1ref. 1101: Combined PWM mode 2 - tmr_oc0ref has the same behavior as in PWM mode 2. tmr_oc0refc is the logical AND between tmr_oc0ref and tmr_oc1ref. 1110: Asymmetric PWM mode 1 - tmr_oc0ref has the same behavior as in PWM mode 1. tmr_oc0refc outputs tmr_oc0ref when the counter is counting up, tmr_oc1ref when it is counting down. 1111: Asymmetric PWM mode 2 - tmr_oc0ref has the same behavior as in PWM mode 2. tmr_oc0refc outputs tmr_oc0ref when the counter is counting up, tmr_oc1ref when it is counting down. **NOTE:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in the BDTR register) and CH0MODESEL = '00' (the channel is configured in output). In PWM mode, 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. On channels having a complementary output, this bit field is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register, then the 'CH0OCMODE' active bits take the new value from the preloaded bits only when a COM event is generated. [3]: CH0CCRSHDWEN; CH0 Output Compare shadow enable 0: Preload register on CCR0 disabled. CCR0 can be written at any time; the new value is taken into account immediately. 1: Preload register on CCR0 enabled. Read/write operations access the preload register. CCR0 preload value is loaded in the active register at each update event.

Field	Name	Description
		NOTE: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in the BDTR register) and CH0MODESEL = '00' (the channel is configured in output). The PWM mode can be used without validating the preload register only in one pulse mode ('SINGLEPLSMODE' bit set in CTRL1 register). Else the behavior is not guaranteed. [2]: CH0OCFASTEN; CH0 Output Compare fast enable This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in single-pulse mode ('SINGLEPLSMODE' bit set in CTRL1 register) to have the output pulse starting as soon as possible after the starting trigger. 0: CC1 behaves normally depending on the counter and CCR0 values even when the trigger is on. The minimum delay to activate CH0 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 CH0 output. Then, OC is set to the compare level independently from the result of the comparison. The delay to sample the trigger input and to activate CH0 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode. [1:0]: CH0MODESEL; CH0 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH0 channel is configured as output 01: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_ch0 10: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_ch1 11: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through 'ITRGSEL' bit (SMCFG register) **NOTE:** CH0MODESEL bits are writable only when the channel is off (CH0EN = '0' in CCER). **Input capture mode**: [15:12]: CH1ICFLT; CH1 Input capture filter. Refer to CH0ICFLT. [11:10]: CH1ICPSC; CH1 Input capture prescaler. Refer to CH0ICPSC. [9:8]: CH1MODESEL; CH1 I/O mode selection

Field	Name	Description
		This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH1 channel is configured as output 01: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_ch1 10: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_ch0 11: CH1 channel is configured as input, tmr_ic1 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through the 'ITRGSEL' bit (SMCFG register) **NOTE:** CH1MODESEL bits are writable only when the channel is off (CH1EN = '0' in CCER). [7:4]: CH0ICFLT; CH0 Input capture filter This bitfield defines the frequency used to sample tmr_ti1 input and the length of the digital filter applied to tmr_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: $f_{SAMPLING}$ = fclk_tmr, N = 2 0010: $f_{SAMPLING}$ = fclk_tmr, N = 4 0011: $f_{SAMPLING}$ = fclk_tmr, 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]: CH0ICPSC; CH0 Input capture prescaler

Field	Name	Description
		This bitfield defines the ratio of the prescaler acting on CH0 input (tmr_ic0). The prescaler is reset as soon as CH0EN = '0' (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]: CH0MODESEL: CH0 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH0 channel is configured as output 01: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_ch0 10: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_ch1 11: CH0 channel is configured as input, tmr_ic0 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through 'ITRGSEL' bit (SMCFG register) **NOTE:** CH0MODESEL bits are writable only when the channel is off (CH0EN = '0' in CCER).

18.4.2.8 GPTMR_CHXMODECTRL2

0x001c			TMR capture/compare mode register 2											GPTMR_CHXMODECTRL2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved							CCMR2								
Type	RO							RW								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CCMR2															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-12 TMR capture/compare Mode Register 2 Description

Field	Name	Description
31:25	Reserved	Reserved
24:0	CCMR2	The field of this register will have different functions when the channel is in different modes (input capture/output compare) **Output compare mode**: [24]: CH3OCMODE_bit3; The bit3 of CH3OCMODE [16]: CH2OCMODE_bit3; The bit3 of CH2OCMODE [15]: CH3OCREFCLREN; CH3 Output Compare OCxREF clear enable. Refer to CH2OCREFCLREN. [14:12]: CH3OCMODE; CH3 Output Compare mode selection. Refer to CH2OCMODE. [11]: CH3CCRSHDWEN; CH3 Output Compare shadow enable. Refer to CH2CCRSHDWEN. [10]: CH3OCFASTEN; CH3 Output Compare fast enable. Refer to CH2OCFASTEN. [9:8]: CH3MODESEL; CH3 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH3 channel is configured as output 01: CH3 channel is configured as input, tmr_ic3 is mapped on tmr_ch3

Field	Name	Description
		10: CH3 channel is configured as input, tmr_ic1 is mapped on tmr_ch2 11: CH3 channel is configured as input, tmr_ic1 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through the 'ITRGSEL' bit (SMCFG register) **NOTE:** CH3MODESEL bits are writable only when the channel is off (CH3EN = '0' in CCER). [7]: CH2OCREFCLREN; CH2 Output Compare OCxREF clear enable 0: tmr_oc1ref is not affected by the tmr_ocref_clr_int signal 1: tmr_oc1ref is cleared as soon as a High level is detected on tmr_ocref_clr_int signal (tmr_ocref_clr input or tmr_etrf input) [6:4]: CH2OCMODE; CH2 Output Compare mode selection These bits define the behavior of the output reference signal tmr_oc0ref from which tmr_oc0 and tmr_oc0n are derived. tmr_oc0ref is active high whereas tmr_oc0 and tmr_oc0n active level depends on 'CH2P' and 'CH2NP' bits. 0000: Frozen - The comparison between the output compare register CCR0 and the counter TCNT has no effect on the outputs. (This mode is used to generate a tmring base). 0001: Set channel 0 to active level on match. tmr_oc0ref signal is forced high when the counter TCNT matches the capture/ compare register 0 (CCR0). 0010: Set channel 0 to inactive level on match. tmr_oc0ref signal is forced low when the counter TCNT matches the capture/ compare register 0 (CCR0). 0011: Toggle - tmr_oc0ref toggles when TCNT = CCR0. 0100: Force inactive level - tmr_oc0ref is forced low. 0101: Force active level - tmr_oc0ref is forced high. 0110: PWM mode 1 - In up-counting, channel 0 is active as long as TCNT is less than CCR0 else inactive. In down-counting, channel 0 is inactive (tmr_oc0ref = '0') as long as TCNT is great than CCR0 else active (tmr_oc0ref = '1'). 0111: PWM mode 2 - In up-counting, channel 0 is inactive as long as TCNT less than CCR0 else active. In down-counting, channel 0 is active as long as TCNT is greater than CCR0 else inactive. 1000: Retrigerrable SINGLEPLSMODE mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become inactive at the next update.

Field	Name	Description
		1001: Retrigerrable SINGLEPLSMODE mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 2 and the channels become inactive at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tmr_itrg signal). Then, a comparison is performed as in PWM mode 1 and the channels become active again at the next update. 1010: Reserved 1011: Reserved 1100: Combined PWM mode 1 - tmr_oc0ref has the same behavior as in PWM mode 1. tmr_oc0refc is the logical OR between tmr_oc0ref and tmr_oc1ref. 1101: Combined PWM mode 2 - tmr_oc0ref has the same behavior as in PWM mode 2. tmr_oc0refc is the logical AND between tmr_oc0ref and tmr_oc1ref. 1110: Asymmetric PWM mode 1 - tmr_oc0ref has the same behavior as in PWM mode 1. tmr_oc0refc outputs tmr_oc0ref when the counter is counting up, tmr_oc1ref when it is counting down. 1111: Asymmetric PWM mode 2 - tmr_oc0ref has the same behavior as in PWM mode 2. tmr_oc1refc outputs tmr_oc0ref when the counter is counting up, tmr_oc1ref when it is counting down. **NOTE:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in the BDTR register) and CH0MODESEL = '00' (the channel is configured in output). In PWM mode, 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. On channels having a complementary output, this bit field is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register then the 'CH2OCMODE' active bits take the new value from the preloaded bits only when a COM event is generated. [3]: CH2CCRSHDWEN; CH2 Output Compare shadow enable 0: Preload register on CCR0 disabled. CCR0 can be written at any time; the new value is taken into account immediately. 1: Preload register on CCR0 enabled. Read/write operations access the preload register. CCR0 preload value is loaded in the active register at each update event. Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in the BDTR register) and ch0_mode_sel='00' (the channel is configured in output).

Field	Name	Description
		The PWM mode can be used without validating the preload register only in one pulse mode ('SINGLEPLSMODE' bit set in CTRL1 register). Else the behavior is not guaranteed. [2]: CH2OCFASTEN; CH2 Output Compare fast enable This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in single-pulse mode ('SINGLEPLSMODE' bit set inCTRL1register), to have the output pulse starting as soon as possible after the starting trigger. 0: CC1 behaves normally depending on the counter and CCR0 values even when the trigger is on. The minimum delay to activate CH2 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 CH2 output. Then, OC is set to the compare level independently from the result of the comparison. The delay to sample the trigger input and to activate CH2 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode. [1:0]: CH2MODESEL; CH2 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH2 channel is configured as output 01: CH2 channel is configured as input, tmr_ic0 is mapped on tmr_ch2 10: CH2 channel is configured as input, tmr_ic0 is mapped on tmr_ch3 11: CH2 channel is configured as input, tmr_ic0 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through 'ITRGSEL' bit (SMCFG register) **NOTE:** CH2MODESEL bits are writable only when the channel is off (CH2EN = '0' in CCER). **Input capture mode**: [15:12]: CH3ICFLT; CH3 Input capture filter. Refer to CH2ICFLT. [11:10]: CH3ICPSC; CH3 Input capture prescaler. Refer to CH2ICPSC. [9:8]: CH3MODESEL; CH3 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH3 channel is configured as output 01: CH3 channel is configured as input, tmr_ic3 is mapped on tmr_ch3 10: CH3 channel is configured as input, tmr_ic1 is mapped on tmr_ch2

Field	Name	Description
		11: CH3 channel is configured as input, tmr_ic1 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through the 'ITRGSEL' bit (SMCFG register) **NOTE:** CH3MODESEL bits are writable only when the channel is off (CH3EN = '0' in CCER). [7:4]: CH2ICFLT; CH2 Input capture filter This bitfield defines the frequency used to sample tmr_ti1 input and the length of the digital filter applied to tmr_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: $f_{SAMPLING}$ = fclk_tmr, N = 2 0010: $f_{SAMPLING}$ = fclk_tmr, N = 4 0011: $f_{SAMPLING}$ = fclk_tmr, 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]: CH2ICPSC; CH2 Input capture prescaler This bitfield defines the ratio of the prescaler acting on CH2 input (tmr_ic0). The prescaler is reset as soon as CH2EN = '0' (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

Field	Name	Description
		10: Capture is done once every 4 events 11: Capture is done once every 8 events [1:0]: CH2MODESEL:CH2 I/O mode selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CH2 channel is configured as output 01: CH2 channel is configured as input, tmr_ic0 is mapped on tmr_ch2 10: CH2 channel is configured as input, tmr_ic0 is mapped on tmr_ch3 11: CH2 channel is configured as input, tmr_ic0 is mapped on tmr_trc. This mode works only if an internal trigger input is selected through 'ITRGSEL' bit (SMCFG register) **NOTE:** CH2MODESEL bits are writable only when the channel is off (CH2EN = '0' in CCER).

18.4.2.9 GPTMR_CCER

0x0020			TMR capture/compare enable register											GPTMR_CCER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH3NP	CH3NEN	CH3P	CH3EN	CH2NP	CH2NEN	CH2P	CH2EN	CH1NP	CH1NEN	CH1P	CH1EN	CH0NP	CH0NEN	CH0P	CH0EN
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-13 TMR capture/compare Enable Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	CH3NP	CH3 capture/compare complementary output polarity. Refer to CH0NP description.
14	CH3NEN	CH3 capture/compare complementary output enable. Refer to CH0NEN description.
13	CH3P	CH3 capture/compare polarity. Refer to CH0P description.
12	CH3EN	CH3 capture/compare enable. Refer to CH0EN description.
11	CH2NP	CH2 capture/compare complementary output polarity. Refer to CH0NP description.
10	CH2NEN	CH2 capture/compare complementary output enable.

Field	Name	Description
		Refer to CH0NEN description.
9	CH2P	CH2 capture/compare polarity. Refer to CH0P description.
8	CH2EN	CH2 capture/compare enable. Refer to CH0EN description.
7	CH1NP	CH1 capture/compare complementary output polarity. Refer to CH0NP description.
6	CH1NEN	CH1 capture/compare complementary output enable. Refer to CH0NEN description
5	CH1P	CH1 capture/compare polarity. Refer to CH0P description.
4	CH1EN	CH1 capture/compare enable. Refer to CH0EN description.
3	CH0NP	CH0 capture/compare complementary output polarity CH0 channel configured as output: 0: tmr_oc0n active high. 1: tmr_oc0n active low. CH0 channel configured as input: This bit is used in conjunction with CH0P to define the polarity of tmr_ch0fp0 and tmr_ch1fp0. Refer to CH0P description. **NOTE:** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in the BDTR register) and CH0MODESEL = '00' (channel configured as output). On channels having a complementary output, this bit is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register then the CH0NP active bit takes the new value from the preloaded bit only when a commutation event is generated.
2	CH0NEN	CH0 capture/compare complementary output enable 0: Off - tmr_oc0n is not active. tmr_oc1n level is then function of MOE, OSSI, OSSR, OIS0, OIS0N and CH0EN bits.

Field	Name	Description
		1: On - tmr_oc0n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS0, OIS0N and CH0EN bits. **NOTE:** On channels having a complementary output, this bit is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register, then the CH0NEN active bit takes the new value from the preloaded bit only when a commutation event is generated.
1	CH0P	CH0 capture/compare polarity 0: OC0 active high (output mode) / Edge sensitivity selection (input mode, see below) 1: OC0 active low (output mode) / Edge sensitivity selection (input mode, see below) When CH0 channel is configured as input, both CH0NP/CH0P bits select the active polarity of CH0FP0 and CH1FP0 for trigger or capture operations. **CH0NP = 0, CH0P = 0**: non-inverted/rising edge. The circuit is sensitive to CHxFP0 rising edge(capture or trigger operations in reset, external clock or trigger mode), CHxFP0 is not inverted (trigger operation in gated mode or encoder mode). **CH0NP = 0, CH0P = 1**: inverted/falling edge. The circuit is sensitive to CHxFP0 falling edge(capture or trigger operations in reset, external clock or trigger mode), CHxFP0 is inverted (trigger operation in gated mode or encoder mode). **CH0NP = 1, CH0P = 1**: non-inverted/both edges/ The circuit is sensitive to both CHxFP0 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), CHxFP0 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode. **CH0NP = 1, CH0P = 0**: the configuration is reserved, it must not be used. **NOTE:** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in the BDTR register). On channels having a complementary output, this bit is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register then the CH0P active bit takes the new value from the preloaded bit only when a commutation event is generated.
0	CH0EN	CH0 capture/compare enable 0: Capture mode disabled/OC0 is not active (see below) 1: Capture mode enabled/OC0 signal is output on the corresponding output pin When CH0 channel is configured as output, the OC0 level depends on MOE, OSSI, OSSR, OIS0, OIS0N and CH0N_EN bits, regardless of the CH0EN bits state. **NOTE:** On channels having a complementary output, this bit is preloaded. If the 'CCSHWDEN' bit is set in the CTRL2 register then the CH0EN active bit takes the new value from the preloaded bit only when a commutation event is generated.

18.4.2.10 GPTMR_TCNT

0x0024			TMR counter register											GPTMR_TCNT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	UPDIF CPY	TCNTH														
Type	RW	RW														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TCNTL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-14 TMR Counter Register Description

Field	Name	Description
31	UPDIFCPY	TCNT[31] or UIFCPY: Value depends on UPDIFREMAP bit in the CTRL1. If UPDIFREMAP = 0 TCNT[31]: Most significant bit of counter value If UPDIFREMAP = 1 UIFCPY: UPDIF copy This bit is a read-only copy of the UPDIF bit of the INTSTS register.
30:16	TCNTH	The register holds the Counter value [30:16] Depends on the bitwidth of TCNT, 16-bit Timer do not have this field.
15:0	TCNTL	The register holds the Counter value [15:0]

18.4.2.11 GPTMR_PSC

0x0028			TMR prescaler register											GPTMR_PSC		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PSC															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-15 TMR Prescaler Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	PSC	Prescaler value The counter clock frequency ($f_{tmr\_cnt\_ck}$) is equal to $f_{tmr\_psc\_ck}$ /(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 'SWUPDGEN' bit of the SWEVTGEN register or through the trigger controller when configured in reset mode).

18.4.2.12 GPTMR_ARR

0x002c			TMR auto-reload register											GPTMR_ARR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	ARRH															
Type	WR															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ARRL															
Type	RW															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 18-16 TMR Auto-reload Register Description

Field	Name	Description
31:16	ARRH	Auto-reload register value MSB Depends on the bitwidth of TCNT, 16-bit Timer do not have this field. ARR is the value to be loaded in the actual auto-reload register.
15:0	ARRL	Auto-reload register value LSB ARR is the value to be loaded in the actual auto-reload register.

18.4.2.13 GPTMR_RCR

0x0030			TMR repetition counter register											GPTMR_RCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	REPCNT															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	REPVAL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-17 TMR Repetition Counter Register Description

Field	Name	Description
31:16	REPCNT	Repetition counter real value
15:0	REPVAL	Repetition counter reload value This bitfield defines the update rate of the compare registers (that is, periodic transfers from preload to active registers) when preload registers are enabled. It also defines the update interrupt generation rate if this interrupt is enabled. When the repetition down-counter reaches zero, an update event is generated and it restarts counting from the REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the 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

18.4.2.14 GPTMR_CCR0

0x0034			TMR capture/compare register 0											GPTMR_CCR0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	CH0CCRVALH															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH0CCRVALL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-18 TMR capture/compare Register 0 Description

Field	Name	Description
31:16	CH0CCRVALH	CH0 capture/compare value MSB Depends on the bitwidth of TCNT, 16-bit Timer do not have this field.
15:0	CH0CCRVALL	CH0 capture/compare value LSB **If channel CH0 is configured as output**: CCR0 is the value to be loaded in the actual capture/compare 0 register (preload value). It is loaded permanently if the preload feature is not selected in the CHxMODECTRL1 register (bit CH0CCRSHDWEN). Else the preload value is copied in the active capture/compare 0 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TCNT and signaled on tmr_oc0 output. **If channel CH0 is configured as input**: CCR0 is the counter value transferred by the last input capture 0 event (tmr_ic0). The CCR0 register is read-only and cannot be programmed.

18.4.2.15 GPTMR_CCR1

0x0038			TMR capture/compare register 1											GPTMR_CCR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	CH1CCRVALH															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH1CCRVALL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-19 TMR capture/compare Register 1 Description

Field	Name	Description
31:16	CH1CCRVALH	CH1 capture/compare value MSB. Depending on the bit width of TCNT, the 16-bit Timer does not have this field.
15:0	CH1CCRVALL	CH1 capture/compare value LSB. Refer to CCR0.

18.4.2.16 GPTMR_CCR2

0x003c			TMR capture/compare register 2											GPTMR_CCR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	CH2CCRVALH															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH2CCRVALL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-20 TMR Capture/compare Register 2 Description

Field	Name	Description
31:16	CH2CCRVALH	CH2 capture/compare value MSB. Depending on the bit width of TCNT, the 16-bit Timer does not have this field.
15:0	CH2CCRVALL	CH2 capture/compare value LSB. Refer to CCR0.

18.4.2.17 GPTMR_CCR3

0x0040			TMR capture/compare register 3											GPTMR_CCR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	CH3CCRVALH															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH3CCRVALL															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-21 TMR Capture/compare Register 3 Description

Field	Name	Description
31:16	CH3CCRVALH	CH3 capture/compare value MSB. Depending on the bit width of TCNT, the 16-bit Timer does not have this field.
15:0	CH3CCRVALL	CH3 capture/compare value LSB. Refer to CCR0.

18.4.2.18 GPTMR_BDTR

0x0044			TMR break and dead-time register											GPTMR_BDTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved												BRKFILTCFG			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MOE	AOE	BRKP	BRKEN	OSSR	OSSI	LOCK		DTG							
Type	RW	RW	RW	RW	RW	RW	RW		RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-22 TMR Break and Dead-time Register Description

Field	Name	Description
31:20	Reserved	Reserved
19:16	BRKFILTCFG	Break filter This bitfield defines the frequency used to sample tmr_brk input and the length of the digital filter applied to tmr_brk. 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, tmr_brk acts asynchronously 0001: $f_{SAMPLING}$ = fclk_tmr, N = 2 0010: $f_{SAMPLING}$ = fclk_tmr, N = 4 0011: $f_{SAMPLING}$ = fclk_tmr, 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

Field	Name	Description
		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 **NOTE:** This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in the BDTR register).
15	MOE	Main output enable This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tmr_brk or tmr_brk2). It is set by software or automatically depending on the AOE bit. It acts only on the channels which are configured in output. 0: In response to a break 2 event. OC and OCN outputs are disabled In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit. 1: OC and OCN outputs are enabled if their respective enable bits are set (CHxEN, CHxNEN in CCER register).
14	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 none of the break inputs tmr_brk and tmr_brk2 is active) **NOTE:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
13	BRKP	Break polarity 0: Break input tmr_brk is active low 1: Break input tmr_brk is active high

Field	Name	Description
		NOTE: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register). Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
12	BRKEN	Break enable This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources). 0: Break function disabled 1: Break function enabled **NOTE:** This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in the BDTR register). Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
11	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. 0: When inactive, OC/OCN outputs are disabled (the timer releases the output control, which is taken over by the GPIO logic, which forces a Hi-Z state). 1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CHxEN = 1 or CHxNEN = 1 (the output is still controlled by the timer). **NOTE:** This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in the BDTR register).
10	OSSI	Off-state selection for idle mode This bit is used when MOE = 0 due to a break event or by a software write on channels configured as outputs. 0: When inactive, OC/OCN outputs are disabled (the timer releases the output control, which is taken over by the GPIO logic and imposes a Hi-Z state).

Field	Name	Description
		1: When inactive, OC/OCN outputs are first forced with their inactive level and then forced to their idle level after the dead-time. The timer maintains its control over the output. **NOTE:** This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in the BDTR register).
9:8	LOCK	Lock configuration These bits provide write protection against software errors. 00: LOCK OFF - No bit is write protected. 01: LOCK Level 1 = DTG bits in BDTR register, OISx and OISxN bits in CTRL2 register and BRKEN/BRKP/AOE bits in BDTR register can no longer be written. 10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CHxP/CHxNP bits in CCER register, as long as the related channel is configured in output through the 'CHxMODESEL' bits) as well as OSSR and OSSI bits can no longer be written. 11: LOCK Level 3 = LOCK Level 2 + CC Control bits (CHxOCMODE and OCxPE bits in CHxMODECTRLx registers, as long as the related channel is configured in output through the 'CHxMODESEL' bits) can no longer be written. **NOTE:** The LOCK bits can be written only once after the reset. Once the BDTR register has been written, their content is frozen until the next reset.
7:0	DTG	Dead-time generator setup This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT corresponds 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])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$. DTG[7:5] = 110 — DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$. DTG[7:5] = 111 — DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$. For example, if TDTS = 125ns (8 MHz), dead-time possible values are: 0 to 15875ns by 125 ns steps 16us to 31750ns by 250 ns steps

Field	Name	Description
		32us to 63us by 1us steps 64us to 126us by 2us steps **NOTE:** This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the BDTR register).

18.4.2.19 GPTMR_CHXSEL

0x0048			TMR timer input selection register											GPTMR_CHXSEL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved					CH3SRCSEL			Reserved					CH2SRCSEL		
Type	RO					RW			RO					RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					CH1SRCSEL			Reserved					CH0SRCSEL		
Type	RO					RW			RO					RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-23 TMR Timer Input Selection Register Description

Field	Name	Description
31:27	Reserved	Reserved
26:24	CH3SRCSEL	Selects tmr_ch3[0..7] input 000: tmr_ch3_in0: TMR_CH3(PAD) 001: tmr_ch3_in1 ... 111: tmr_ch3_in7
23:19	Reserved	Reserved
18:16	CH2SRCSEL	Selects tmr_ch2[0..7] input 000: tmr_ch2_in0: TMR_ch2(PAD) 001: tmr_ch2_in1 ... 111: tmr_ch2_in7

Field	Name	Description
15:11	Reserved	Reserved
10:8	CH1SRCSEL	Selects tmr_ch1[0..7] input 000: tmr_ch1_in0: TMR_ch1(PAD) 001: tmr_ch1_in1 ... 111: tmr_ch1_in7
7:3	Reserved	Reserved
2:0	CH0SRCSEL	Selects tmr_ch0[0..7] input 000: tmr_ch0_in0: tmr_ch0(PAD) 001: tmr_ch0_in1 ... 111: tmr_ch0_in7

18.4.2.20 GPTMR_AF1

0x004c			TMR alternate function option register 1											GPTMR_AF1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								SYSBRKEN				Reserved	ETRGSEL		
Type	RO								RW				RO	RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				BRKCOMP3P	BRKCOMP2P	BRKCOMP1P	BRKINP	Reserved				BRKCOMP3EN	BRKCOMP2EN	BRKCOMP1EN	BRKINEN
Type	RO				RW	RW	RW	RW	RO				RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 18-24 TMR Alternate Function Option Register 1 Description

Field	Name	Description
31:24	Reserved	Reserved
23:20	SYSBRKEN	System Break enable [0]: sbrkin[0] enable,high active [1]: sbrkin[1] enable,high active [2]: sbrkin[2] enable,high active [3]: sbrkin[3] enable,high active **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
19:19	Reserved	Reserved
18:16	ETRGSEL	etr_in source selection These bits select the etr_in input source. 000: tmr_etr0: TMR_ETRG input

Field	Name	Description
		001: tmr_etr1 ... 111: tmr_etr7 **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
15:12	Reserved	Reserved
11	BRKCOMP3P	tmr_brk_cmp3 input polarity This bit selects the tmr_brk_cmp3 input sensitivity. It must be programmed together with the BRKP polarity bit. 0: tmr_brk_cmp3 input polarity is not inverted (active low if BRKP = 0, active high if BRKP = 1) 1: tmr_brk_cmp3 input polarity is inverted (active high if BRKP = 0, active low if BRKP = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
10	BRKCOMP2P	tmr_brk_cmp2 input polarity This bit selects the tmr_brk_cmp2 input sensitivity. It must be programmed together with the BRKP polarity bit. 0: tmr_brk_cmp2 input polarity is not inverted (active low if BRKP = 0, active high if BRKP = 1) 1: tmr_brk_cmp2 input polarity is inverted (active high if BRKP = 0, active low if BRKP = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
9	BRKCOMP1P	tmr_brk_cmp1 input polarity This bit selects the tmr_brk_cmp1 input sensitivity. It must be programmed together with the BRKP polarity bit. 0: tmr_brk_cmp1 input polarity is not inverted (active low if BRKP = 0, active high if BRKP = 1) 1: tmr_brk_cmp1 input polarity is inverted (active high if BRKP = 0, active low if BRKP = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
8	BRKINP	TMR_BKIN input polarity This bit selects the TMR_BKIN alternate function input sensitivity. It must be programmed together with the BRKP polarity bit. 0: TMR_BKIN input polarity is not inverted (active low if BRKP = 0, active high if BRKP = 1)

Field	Name	Description
		1: TMR_BKIN input polarity is inverted (active high if BRKP = 0, active low if BRKP = 1) **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
7:4	Reserved	Reserved
3	BRKCOMP3EN	tmr_brk_cmp3 enable This bit enables the tmr_brk_cmp3 for the timer's tmr_brk input. tmr_brk_cmp3 output is 'ORed' with the other tmr_brk sources. 0: tmr_brk_cmp3 input disabled 1: tmr_brk_cmp3 input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
2	BRKCOMP2EN	tmr_brk_cmp2 enable This bit enables the tmr_brk_cmp2 for the timer's tmr_brk input. tmr_brk_cmp2 output is 'ORed' with the other tmr_brk sources. 0: tmr_brk_cmp2 input disabled 1: tmr_brk_cmp2 input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
1	BRKCOMP1EN	tmr_brk_cmp1 enable This bit enables the tmr_brk_cmp1 for the timer's tmr_brk input. tmr_brk_cmp1 output is 'ORed' with the other tmr_brk sources. 0: tmr_brk_cmp1 input disabled 1: tmr_brk_cmp1 input enabled **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
0	BRKINEN	TMR_BKIN input enable This bit enables the TMR_BKIN alternate function input for the timer's tmr_brk input. TMR_BKIN input is 'ORed' with the other tmr_brk sources. 0: TMR_BKIN input disabled 1: TMR_BKIN input enabled

Field	Name	Description
		NOTE: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).

18.4.2.21 GPTMR_AF2

0x0050			TMR alternate function option register 2											GPTMR_AF2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved														OCREFCLRSRCSEL	
Type	RO														RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-25 TMR Alternate Function Option Register 2 Description

Field	Name	Description
31:18	Reserved	Reserved
17:16	OCREFCLRSRCSEL	ocref_clr source selection These bits select the ocref_clr input source. 00: tmr_ocref_clr0 01: tmr_ocref_clr1 ... 11: tmr_ocref_clr3 **NOTE:** These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in the BDTR register).
15:0	Reserved	Reserved

18.4.2.22 GPTMR_DMACFG

0x0054			TMR DMA control register											GPTMR_DMACFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DBL						Reserved		DBA					
Type	RO		RW						RO		RW					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-26 TMR DMA Control Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:8	DBL	DMA burst length This 6-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or write access is done to the DMA address) 000000: 1 transfer 000001: 2 transfers 000010: 3 transfers ... 010001: 18 transfers
7:6	Reserved	Reserved
5:0	DBA	DMA base address This 6-bit vector defines the base address for DMA transfers (when read/write access is done through the DMAR address). DBA is defined as an offset starting from the address of the CTRL1 register.

Field	Name	Description
		Example: 00000: CTRL1 00001: CTRL2 00010: SMCFG

18.4.2.23 GPTMR_DMAR

0x0058			TMR DMA address for full transfer											GPTMR_DMAR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DMAB															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DMAB															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-27 TMR DMA Address for Full Transfer Description

Field	Name	Description
31:0	DMAB	DMA register for burst accesses A read or write operation to the DMA register accesses the register located at the address (CTRL1 address) + (DBA + DMA index) x 4 where: CTRL1 address is the address of the control register 1. DBA is the DMA base address configured in DMACFG register. DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in DMACFG).

Independent Watchdog (IWDG)

This chapter describes the details of the Independent Watchdog (IWDG).

Topics: **Page**

19.1 Introduction

The Independent Watchdog (IWDG) block can be used for detecting and solving malfunctions caused by software failure. It triggers a system reset when the counter reaches a specified timeout value. The IWDG is clocked by its own dedicated internal low-speed clock (ILS), allowing it to remain active even if the main clock is unavailable.

Furthermore, the IWDG can function as a Window Watchdog by configuring the appropriate window in the WIN bit of IWDG window register. If the reload occurs while the counter exceeds the value stored in the IWDG window register (WIN bit), a reset is triggered.

The IWDG is particularly suitable for applications requiring an Independent Watchdog process separate from the main application, but with lower timing accuracy requirements.

19.2 Features

- Free-running down-counter
- Clocked from an independent RC oscillator internal low-speed clock (ILS) and low frequency oscillator (ELS), which can operate in stop and standby modes.
- The hardware watchdog feature is enabled through the device option bits; the watchdog is automatically enabled at power-on and generates a reset.
- The Independent Watchdog (IWDG) reset flag is logged in the reset and clock control/status register.
- Set IWDG counting and not counting in the Flash option register (FMC_OPTR).
- Register access protection for IWDG
- Programmable 12-bit down-counter
- Can be configured as a Window Watchdog (option)
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x000.
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window.
- IWDG can be enabled or disabled during the system stop and standby modes.
- The IWDG counter either continues to work or stop under debug mode.

19.3 Functional Description

19.3.1 Block Diagram

Figure 19-1 shows the functional blocks of the Independent Watchdog (IWDG) module.

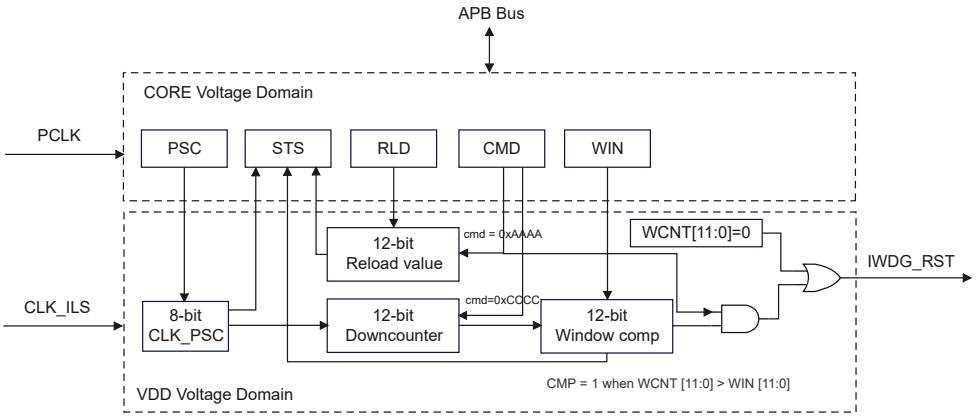


Figure 19-1 IWDG Block Diagram

The internal RC low-speed clock (ILS) acts as a low-power clock source that can be kept running in stop and standby mode for the IWDG. The IWDG Register is in the V_{CORE} domain, while the IWDG counter is in the V_{DD} domain.

The register and counter are in the different clock domains; the register will be activated later after the register is configured with the synchronization mechanism. Hence, it needs to wait for a synchronization time after the previous register is activated, and then configure the next register if the user would like to configure several registers continuously. There are the flag bits that are used to check the status update of the synchronization mechanism in the IWDG status register (IWDG_STS).

19.3.2 Register Access Protection

Write access to the IWDG prescaler configuration register (IWDG_PSC), IWDG reload register (IWDG_RLD), and IWDG window register (IWDG_WIN) is protected. It can avoid unexpectedly modified on these registers.

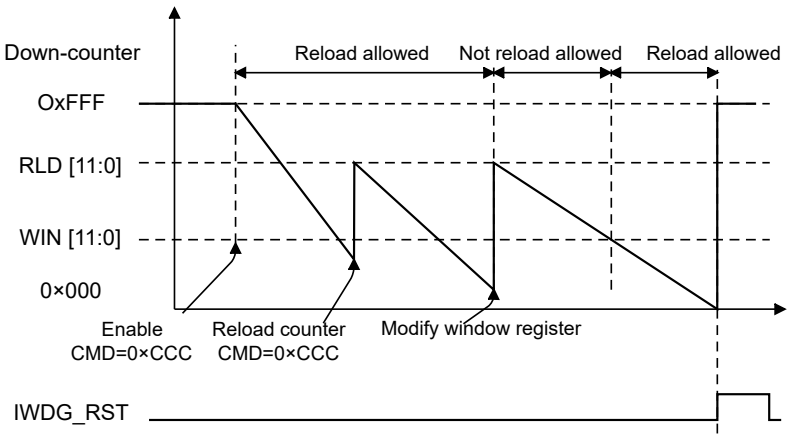
To enable and disable the register access protection, it is only controlled by writing the IWDG command register (IWDG_CMD). The register access protection is enabled by default. The write operation is not working under register access protection mode. To modify them, the user must first write the code 0x0000 5555 in the IWDG_CMD, and then the register access protection is disabled.

A write access to this register with a different value breaks the sequence, and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

19.3.3 Controlling Down-Counter

To set the value 0xCCCC to the CMD[15:0] bits of the IWDG window register (IWDG_WIN), the software can activate the IWDG. Once enabled, the counter will begin counting down from the initial value of 0xFFF. Upon reaching the count value of 0x000, a reset signal is triggered, resulting in an IWDG reset.

The maximum time interval between two reload operations is determined by the RLD[11:0] bits of the IWDG reload register (IWDG_RLD). The down-counter depends on the internal low-speed clock (ILS) and the prescaler divider.



The IWDG can also work as a Window Watchdog by setting the appropriate window WIN[11:0] bits in the IWDG_WIN.

The default value of the WIN[11:0] bits is 0xFFF, so if it is not updated, the window option is disabled. When the WIN[11:0] bits in IWDG_WIN are not 0xFFF, the window option is enabled.

When the window value changes, a reload operation is performed to reset the down-counter to the RELOAD[11:0] value of the IWDG_RLD and ease the cycle number calculation to generate the next reload.

After the window option is enabled, while the WIN[11:0] bits of the IWDG_WIN are used to set the high limit of the window, the down-counter must be reloaded to prevent a reset when its value is lower than the window register value and greater than 0x0; otherwise, the watchdog generates a reset when the down-counter is reloaded outside the window. The value of the WIN[11:0] bits in IWDG_WIN should be from 0xFFF to 0x0.

19.3.4 Calculating IWDG Timeout

To calculate the IWDG timeout, use the following equation:

$$t_{IWDG} = T_{ILS} \times \left(4 \times 2^{PSC[2:0]}\right) \times (RELOAD[11:0] + 1) \quad (5)$$

Where:

- t_{IWDG}: IWDG timeout.
- T_{ILS}: Internal low-speed clock (ILS), ILS clock frequency is 32 kHz.
- $4 \times 2^{PSC[2:0]}$: The value corresponding to the prescaler divider.

As an example, if the ILS clock frequency is 32 kHz, PSC[2:0] is set to 3, and RELOAD[11:0] is set to 0xFFF: t_{IWDG} = 1/32 × 4 × 2^3 × 4096 = 4096.00 ms

Refer to Table 19-1 for the minimum and maximum values of $t_{IWDG}$when ILS is 32 kHz.

Table 19-1 Min/Max IWDG Timeout Period (in ms) at 32 kHz (ILS)

PSC[2:0]	Prescaler	Min Timeout Value RELOAD[11:0] = 0x000	Max Timeout Value RELOAD[11:0] = 0xFFF
3'b000	4	0.125	512.00
3'b001	8	0.25	1024.00
3'b010	16	0.50	2048.00
3'b011	32	1.00	4096.00
3'b100	64	2.00	8192.00
3'b101	128	4.00	16384.00
3'b110 or 3'b111	256	8.00	32768.00

19.3.5 Enabling IWDG

The user can select the option for Window Watchdog enabled by the hardware or software; it can be set in the IWDGSW bit Independent Watchdog selection of Flash option byte organization.

When the user option IWDGSW = 0 selects Hardware Independent Watchdog, the watchdog is always enabled after a reset, it cannot be disabled.

When the user option IWDGSW = 1 selects Software Independent Watchdog, the watchdog is always disabled after a reset. The IWDG is started by writing the value 0x0000 CCCC in the IWDG command register (CMD). Once it is started, it cannot be stopped except by a reset. The software Window Watchdog is in default.

19.3.6 Enabling Internal Low-Speed Clock

The IWDG clock is clocked by the internal low-speed clock (ILS) or low-frequency oscillator (ELS). The ILS RC acts as a low-power clock source that can be kept running in stop and standby mode for the IWDG and RTC. The clock frequency is 32 kHz.

The ILS RC can be switched on and off using the bits in the RCC domain control register (RCC_BDCR). After the system is powered on, an ILS is generated.

For more information on the enable and ready selection of the ILS oscillator, refer to the description of the ILSON and ILSRDY bits in RCC_BDCR.

19.3.7 IWDG Reset Flag

The Independent Watchdog reset flag (IWDGRSTF) is recorded in the RCC control/status register (RCC_CSR). The IWDGRSTF bit is set by the hardware in the event of an IWDG reset.

For more information, refer to the description of the IWDGRSTF bit description in RCC_CSR.

19.3.8 IWDG Enable in Low-Power Mode

The IWDG can be enabled or disabled in the system standby and stop modes. The IWDGSTDBY and IWDGSTOP bits in the Flash option register (FMC_OPTR) can be configured to control the IWDG.

For more details, refer to the description of the IWDGSTDBY and IWDGSTOP bits in FMC_OPTR.

19.3.9 Debug Mode

When the device enters debug mode and the core is halted, the IWDG counter may either continue to function normally or stop altogether. This is dependent on the configuration of the corresponding bit DBGIWDGSTOP in the DBG module.

The DBGIWDGSTOP bit is set in the Debug MCU APB freeze register (DBGMCU_APBFZR). Refer to the DBGWWDGSTOP bit description in DBGMCU_APBFZR for more details.

19.4 Registers

19.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	IWDG_CMD	IWDG command register
0x0004	IWDG_PSC	IWDG prescaler configuration register
0x0008	IWDG_RLD	IWDG reload register
0x000c	IWDG_STS	IWDG status register
0x0010	IWDG_WIN	IWDG window register
0x0014	IWDG_CLKSEL	IWDG cock source selection register

19.4.2 Register Field Details

The IWDG peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

19.4.2.1 IWDG_CMD

0x0000			IWDG command register											IWDG_CMD		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CMD															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-2 IWDG Command Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CMD	Command register, readback value is 0 16'h5555: Enable access to the PSC, RLD and WIN register 16'hCCCC: Start the watchdog if IWDGSW = 1 16'hAAAA: Reload the IWDG counter value

19.4.2.2 IWDG_PSC

0x0004			IWDG prescaler configuration register											IWDG_PSC		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved													PSC		
Type	RO													RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-3 IWDG Prescaler Configuration Register Description

Field	Name	Description
31:3	Reserved	Reserved
2:0	PSC	Prescaler divider These bits are written access protected. They are written by software to select the prescaler divider feeding the counter clock. The PSCUPD bit of the IWDG_STS must be 0 to change the prescaler divider. 3'b000: divider /4 3'b001: divider /8 3'b010: divider /16 3'b011: divider /32 3'b100: divider /64 3'b101: divider /128 3'b110: divider /256 3'b111: divider /256

Field	Name	Description
		NOTE: When reading this register, the prescaler value is returned from the VDD voltage domain. The validity of this value may be compromised if a write operation to this register is in progress. Therefore, the value read from this register is considered valid only when the PSCUPD bit in the IWDG_STS is reset.

19.4.2.3 IWDG_RLD

0x0008			IWDG reload register											IWDG_RLD		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				RELOAD											
Type	RO				RW											
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Table 19-4 IWDG Reload Register Description

Field	Name	Description
31:12	Reserved	Reserved
11:0	RELOAD	Watchdog counter reload value These bits are written access protected; they are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the IWDG command register (IWDG_CMD). The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. The RLDUPD bit in the IWDG status register (IWDG_STS) must be 0 to be able to change the reload value. **NOTE:** When reading this register, the prescaler value is returned from the VDD voltage domain. The validity of this value may be compromised if a write operation to this register is in progress. Therefore, the value read from this register is considered valid only when the PSCUPD bit in the IWDG_STS is reset.

19.4.2.4 IWDG_STS

0x000c			IWDG status register											IWDG_STS		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved											RLDTRGUPD	ENBUPD	WINUPD	RLDUPD	PSCUPD
Type	RO											RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-5 IWDG Status Register Description

Field	Name	Description
31:5	Reserved	Reserved
4	RLDTRGUPD	Watchdog reload trigger update This bit is set by hardware to indicate that counter reload (CMD = 16'hAAAA) is ongoing. It is reset by hardware when the counter reload is completed in the VDD voltage domain.
3	ENBUPD	Watchdog enable update This bit is set by hardware to indicate that IWDG enable (CMD = 16'hCCCC) is ongoing. It is reset by hardware when the IWDG enable is high in the VDD voltage domain.
2	WINUPD	Watchdog counter window value update This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the VDD voltage domain.

Field	Name	Description
		Window value can be updated only when the WINUPD bit is reset.
1	RLDUPD	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. Reload value can be updated only when the RLDUPD bit is reset.
0	PSCUPD	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. Prescaler value can be updated only when the PSCUPD bit is reset.

19.4.2.5 IWDG_WIN

0x0010			IWDG window register											IWDG_WIN		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				WIN											
Type	RO				RW											
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Table 19-6 IWDG Window Register Description

Field	Name	Description
31:12	Reserved	Reserved
11:0	WIN	Watchdog counter window value These bits are written access protected; they contain the high limit of the window value to be compared with the down-counter. To prevent a reset, the down-counter must be reloaded when its value is lower than the window register value and greater than 0x0. The WINUPD bit in the IWDG status register (IWDG_STS) must be reset to change the reload value. **NOTE:** When reading this register, the prescaler value is returned from the VDD voltage domain. The validity of this value may be compromised if a write operation to this register is in progress. Therefore, the value read from this register is considered valid only when the PSCUPD bit in the IWDG_STS is reset.

19.4.2.6 IWDG_CLKSEL

0x0014			IWDG cock source selection register											IWDG_CLKSEL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															CLKSEL
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19-7 IWDG Cock Source Selection Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	CLKSEL	IWDG clock source selection 0: ILS 1: ELS

Window Watchdog (WWDG)

This chapter describes the details of the Window Watchdog (WWDG).

Topics: **Page**

20.1 Introduction

The Window Watchdog (WWDG) is used to detect software faults that are typically caused by external interference or unforeseen logical issues, which can lead the application program to enter an abnormal state. Generally, the windows watchdog circuit generates an MCU reset when a programmed period expires unless the program refreshes the contents of the down-counter before the 7-bit counter register is cleared. An MCU reset is also generated if the 7-bit down-counter value (in the WWDG control register) is refreshed before reaching the window register value. This means that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB PCLK and has a configurable time window that can be programmed to detect abnormally late or early application behavior. The WWDG is particularly well-suited for applications that require the watchdog to react within an accurate timing window.

20.2 Features

- Support for keeping active after a reset by setting the Flash option.
- Programmable 7-bit free-running down-counter
- Conditional reset when the watchdog is activated:
  - When the down-counter value becomes lower than 0x40.
  - When the down-counter is reloaded outside the window (refer to Figure 20-2).
- Early Wakeup Interrupt (EWI) can be triggered when the down-counter is equal to 0x40.
- Pause or keep working during a breakpoint in debug mode.

20.3 Functional Description

The RSTEN bit is enabled in the WWDG control register to activate the windows watchdog (WWDG). When the WWDG is activated, and the 7-bit down-counter (WCNT[6:0] bits) is decremented from 0x40 to 0x3F, it will trigger a reset. If the software reloads the counter while the counter is higher than the value stored in the window register, a reset will be generated.

The software needs to periodically write to the WWDG control register (WWDG_CTRL) in order to avoid a reset. This action should occur only when the counter value is less than the window register value and greater than 0x3F. The value stored in the WWDG_CTRL must be within the range of 0xFF and 0xC0.

20.3.1 Block Diagram

Figure 20-1 shows the block diagram of the WWDG.

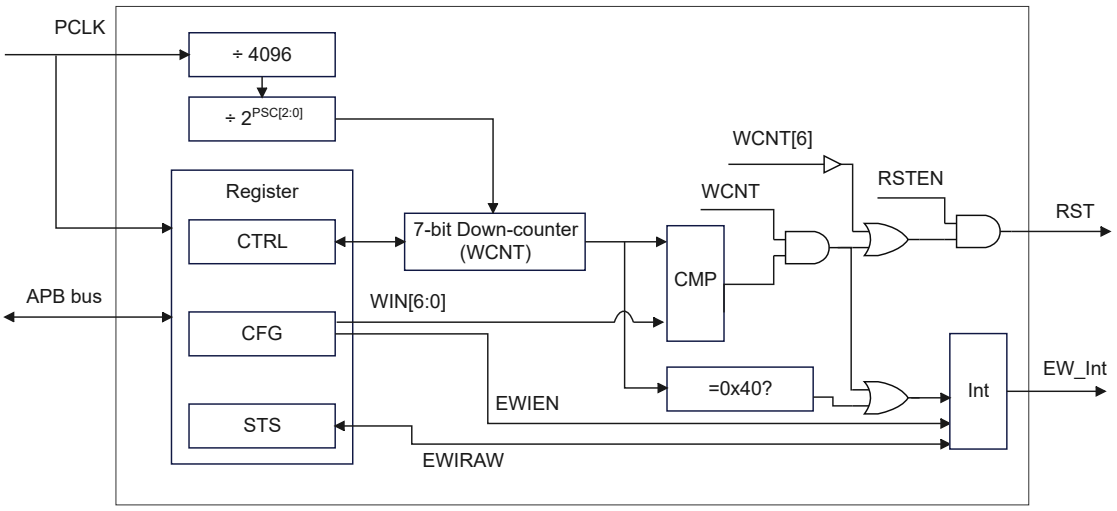


Figure 20-1 WWDG Block Diagram

The WWDG is one of the peripherals from the bus APB; the WWDG clock is prescaled from the APB PCLK.

When the application program is working with the watchdog enabled, always write 1 to the WCNT[6:0] bit of the WWDG control register, which reacts watchdog with an accurate timing window that is to avoid generating an immediate reset. The WIN[6:0] bits of the WWDG configuration register are used to set the high limit of the window.

It is defined as the window period when the counter value is lower than or equal to the window register value (WCNT[6:0] bits) and higher than 0x3F (the low limit of the window). Otherwise, it is outside the window.

20.3.2 Enabling Watchdog Clock

Reset and clock control register also has the clock enable bit for WWDG. After powered on, the WWDG is activated with the Window Watchdog clock enabled on the Window Watchdog clock enable (WWDGEN) bit.

NOTE: WWDGEN bit is set in the APB1 peripheral enable register. Refer to the WWDGEN bit description in RCC_APB1ENR for details.

20.3.3 Enabling WWDG

Whether the Window Watchdog (WWDG) is enabled by the hardware or software can be selected. This selection can be made in the WWDGSW bit of the Flash option byte organization.

When WWDGSW = 0, the hardware Window Watchdog is selected, which remains enabled even after a reset and cannot be disabled.

When WWDGSW = 1, the software Window Watchdog is selected, but it remains disabled after a reset. To enable it, the RSTEN bit in the WWDG configuration register must be set, and once enabled, it cannot be disabled again except by a reset. The software Window Watchdog is the default option.

NOTE: Window Watchdog selection (WWDGSW) bit is reset in the Flash option register (FMC_OPTR).

20.3.4 Controlling Down-Counter

This down-counter operates in a free-running mode, starting from the default value 0x7F once PCLK is stable, regardless of the watchdog being disabled. The software is responsible for reloading the counter (WCNT[6:0] bits) while ensuring that the counter remains within the range of 0x40 to 0x7F. To prevent an immediate reset, the WCNT[6] bit must be set to 1. Additionally, the current value of the counter is stored in this register and can be read back by the software.

The WCNT[6:0] bits contain the number of increments that determine the time delay before the watchdog triggers a reset. The timing range is determined by the configuration of the PCLK and the prescaler when writing to the WWDG control register (WWDG_CTRL). For more details, refer to Figure 20-2.

The WWDG configuration register stores the upper limit of the window within which a reset can be prevented. The down-counter needs to be reloaded when its value falls below the window register value but remains greater than 0x3F. For a detailed description of the WWDG process, refer to Figure 20-2.

NOTE: The WCNT6 bit can be used to trigger a software reset. This can be achieved by setting the RSTEN bit and clearing the WCNT6 bit.

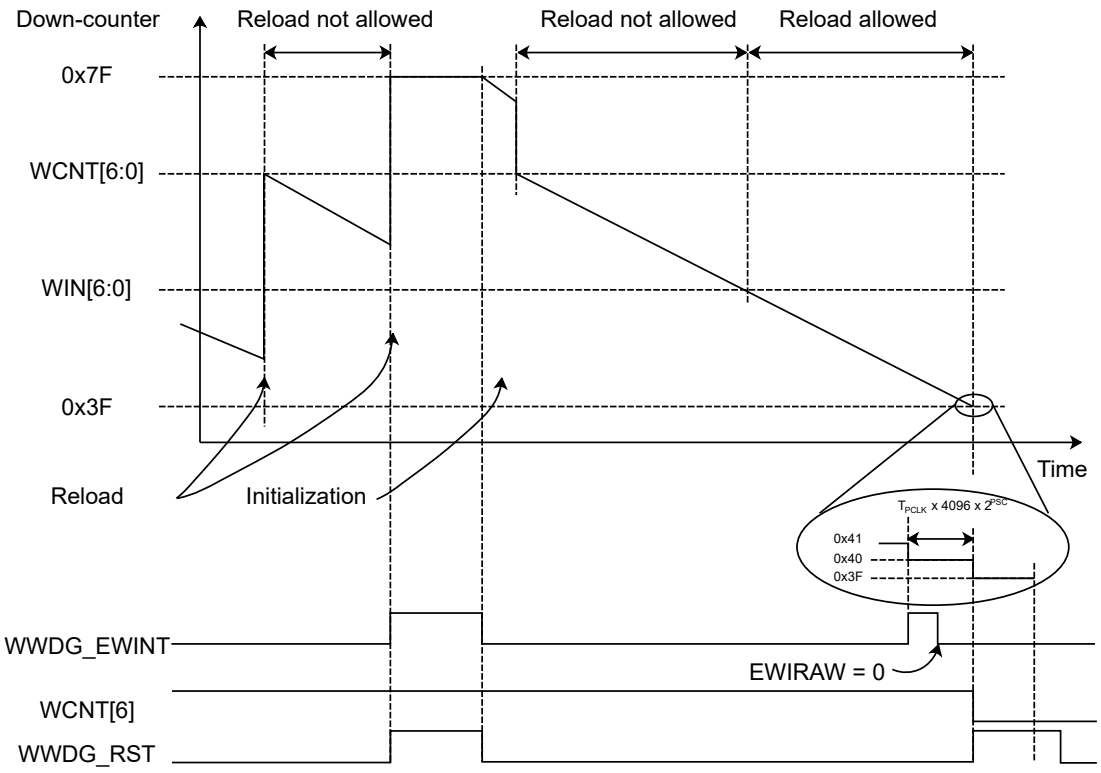


Figure 20-2 WWDG Down-Counter Timing Diagram

If the watchdog reset is enabled (when the RSTEN bit is set to 1 in the WWDG_CTRL), you can periodically update the 7-bit counter value of the WWDG_CTRL to avoid a reset. This update should only be performed when the counter value is less than the window value and greater than 0x3F.

The watchdog reset is triggered when the software refreshes the down-counter while it is greater than the value stored in the window register, which is considered outside the window. Additionally, if enabled and the watchdog is activated, an early wakeup interrupt (EWI) is triggered when the down-counter reaches the value 0x40. Following a watchdog reset, the 7-bit down-counter (WCNT[6:0] bits) is decremented from 0x7F. This process will restart the software initialization for WWDG.

When the down-counter is equal to 0x40, the EWI is triggered if enabled, and the EWI Flag is also set. To clear the EWI interrupt, you need to write '0' to the EWIRAW bit in the WWDG status register (WWDG_STS).

When the down-counter is decremented from 0x40 to 0x3F (WCNT[6] becomes clear), the WWDG initiates a reset.

20.3.5 Calculating WWDG Timeout

To calculate the WWDG timeout, you can use the following equation:

$$t_{WWDG} = T_{PCLK} \times 4096 \times 2^{PSC[2:0]} \times (WCNT[6:0] + 1) \ (ms) \quad (6)$$

Where:

- t_{WWDG}: WWDG timeout.
- T_{PCLK}: APB clock period measured in ms.
- 4096: The value corresponding to the internal divider.

As an example, if APB frequency is 48 MHz, PSC[2:0] is set to 3 and WCNT[5:0] is set to 63: t_{WWDG} = 1/48000 × 4096 × 2^3× 63 + 1 = 43.69 ms

Refer to Table 20-1 for the minimum and maximum values of t_{WWDG} when PCLK is 48 MHz.

Table 20-1 Min/Max WWDG Timeout Period (in ms)

PSC[2:0]	Prescaler	Minimum Timeout Value WCNT[6:0] = 0x40	Maximum Timeout Value WCNT[6:0] = 0x7F
3'b000	1	0.09	5.46
3'b001	2	0.17	10.92
3'b010	4	0.34	21.85
3'b011	8	0.68	43.69
3'b100	16	1.37	87.38
3'b101	32	2.73	174.76
3'b110	64	5.46	349.53
3'b111	128	10.92	699.05

20.3.6 WWDG Software Reset

Reset and Clock Control (RCC) register also has the reset bit for the Window Watchdog (WWDG).

The WWDG reset (WWDGRST) bit is set in the APB1 peripheral reset register (RCC_APB1RSTR). Refer to the WWDGRST bit description in RCC_APB1RSTR for more details.

20.3.7 WWDG Reset Flag

The Window Watchdog Reset Flag (WWDGRSTF) is logged in the RCC Control/Status Register (RCC_CSR). When the WWDG generates a reset, the WWDGRSTF flag is set in the RCC_CSR to identify the reset source.

Refer to the WWDGRSTF bit description in RCC_CSR for more details.

20.3.8 Debug Mode

When the device enters debug mode (core halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in the DBG module.

The DBGWWDGSTOP is set in the Debug MCU APB freeze register (DBGMCU_APBFZR). Refer to the DBGWWDGSTOP bit description in DBGMCU_APBFZR for more details.

20.4 Registers

20.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	WWDG_CTRL	WWDG control register
0x0004	WWDG_CFG	WWDG configuration register
0x0008	WWDG_STS	WWDG status register

20.4.2 Register Field Details

20.4.2.1 WWDG_CTRL

0x0000			WWDG control register											WWDG_CTRL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								RSTEN	WCNT						
Type	RO								RS	RW						
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Table 20-2 WWDG Control Register Description

Field	Name	Description
31:8	Reserved	Reserved
7	RSTEN	WWDG reset enable This bit is set by software and only cleared by hardware after a reset. When RSTEN = 1, the watchdog can generate a reset. 0: Watchdog reset disabled 1: Watchdog reset enabled
6:0	WCNT	7-bit counter (MSB to LSB), these bits contain the value of the watchdog counter, decremented every (4096 x $2^{psc[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (bit6 cleared).

20.4.2.2 WWDG_CFG

0x0004			WWDG configuration register											WWDG_CFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		PSC			Reserved	EWIEN	Reserved		WIN						
Type	RO		RW			RO	RS	RO		RW						
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Table 20-3 WWDG Configuration Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:11	PSC	Timer base 000: The counter clock is (PCLK / 4096) / 1 001: The counter clock is (PCLK / 4096) / 2 010: The counter clock is (PCLK / 4096) / 4 011: The counter clock is (PCLK / 4096) / 8 100: The counter clock is (PCLK / 4096) / 16 101: The counter clock is (PCLK / 4096) / 32 110: The counter clock is (PCLK / 4096) / 64 111: The counter clock is (PCLK / 4096) / 128
10:10	Reserved	Reserved

Field	Name	Description
9	EWIEN	Early wakeup interrupt enable When set, an interrupt occurs whenever the counter reaches the value 0x40 or update the counter value before it matches the window value. This interrupt is only cleared by hardware after a reset.
8:7	Reserved	Reserved
6:0	WIN	7-bit window value These bits contain the upper limit of the window value to be compared with the down-counter; the lower limit value is 0x3F.

20.4.2.3 WWDG_STS

0x0008			WWDG status register											WWDG_STS		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															EWIRAW
Type	RO															RC_W0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-4 WWDG Status Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	EWIRAW	Early wakeup interrupt flag, not affected by EWIEN This bit is set by hardware when the counter has reached the value 0x40, or update the counter value before it matches the window value. It must be cleared by software by writing '0'. Writing '1' has no effect. This bit is also set if the early wakeup interrupt is not enabled.

Real-Time Clock (RTC)

This chapter describes the details of the Real-Time Clock (RTC).

Topics: **Page**

21.1 Introduction

The Real-Time Clock (RTC) provides automatic wakeup functionality to manage all low-power modes. It operates as an independent BCD timer/counter and offers a time-of-day clock/calendar with programmable alarm interrupts.

Regardless of the device status (active mode, low-power mode, or under reset), the RTC continues to function as long as the supply voltage remains within the operating range.

The RTC remains functional in VBAT mode. It includes 16 backup registers that are retained in both low-power modes and VBAT mode. These backup registers, denoted as BKPxR (x=1,2,3,...,16), are implemented in the RTC domain, which remains powered-on by VBAT when the VDD power is switched off.

21.2 Features

- Calendar with subsecond, seconds, minutes, hours (12 or 24-hour format), weekday, date, month, year, in BCD (binary-coded decimal) format.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- Real-time correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to improve the calendar's accuracy.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp function that can be used to save the calendar content. This feature can be triggered by an event on the timestamp pinor by switching to VBAT mode.
- A 17-bit auto-reload wakeup timer for periodic events with programmable resolution and period.

The RTC is supplied through a switch that takes power either from the VDD supply when present or from the VBAT pin.

The RTC clock sources include:

- An external low-speed oscillator (ELS)
- The internal low-speed RC oscillator (ILS, with typical frequency of 32 KHz)
- The external high-speed crystal or oscillator (EHS), divided by a prescaler in the RCC.

The RTC is functional in VBAT mode and in all low-power modes when it's clocked by the ELS.

However, when it's clocked by the ILS, the RTC doesn't function in VBAT mode, but still operates in all low-power modes except for shutdown mode.

Additionally, all RTC events, including Alarm, Wakeup Timer and Timestamp, are capable of generating an interrupt and waking up the device from low-power modes.

21.3 Functional Description

21.3.1 Block Diagram

Figure 21-1 shows the block diagram of the RTC module.

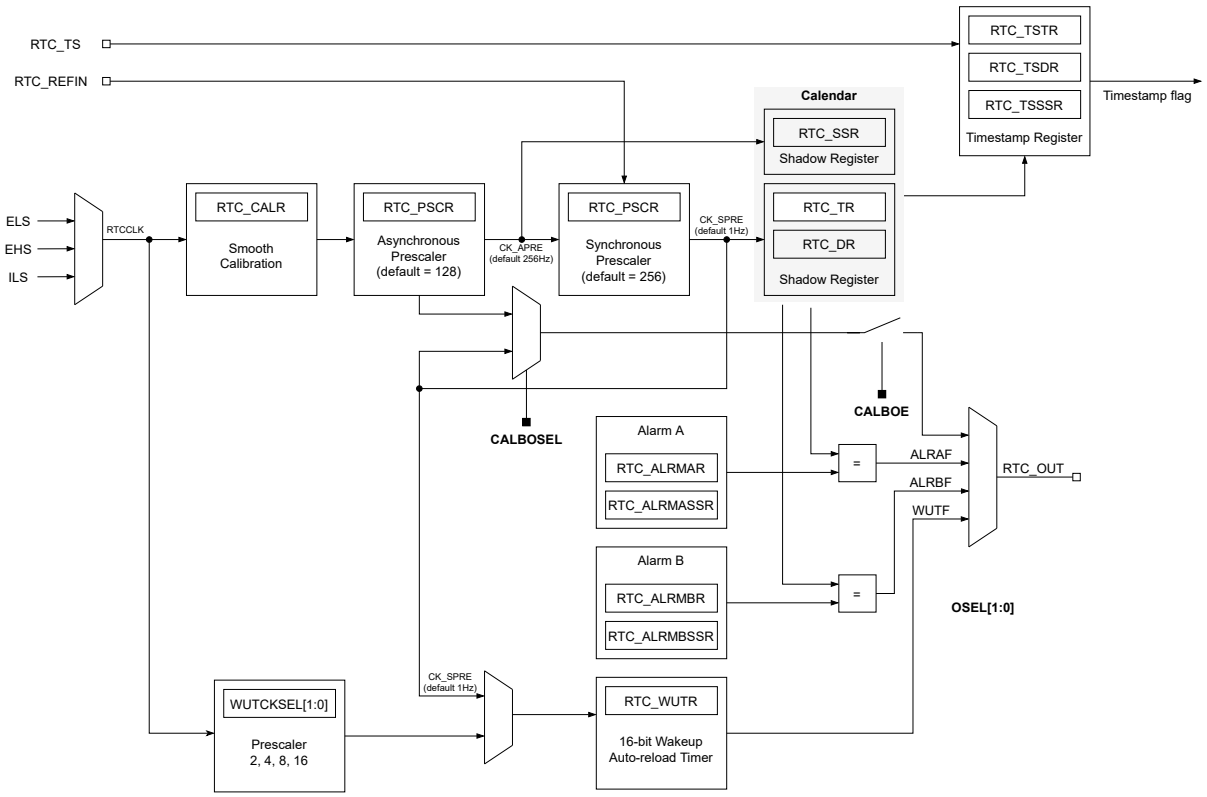


Figure 21-1 Real-Time Clock (RTC) Block Diagram

21.3.2 Pins and Internal Signals

Table 21-1 RTC Input/Output Pins

Pin Name	Signal Type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT	Output	RTC output

RTC_OUT selects one of the following two outputs:

- RTC_CALIB: 512 Hz or 1 Hz clock output (with an ELS frequency of 32.768 KHz). This output is enabled by setting the CALBOE bit in the RTC control register (RTC_CR).
- RTC_ALARM: This output is one of the sources from the wakeup timer, ALARM A and ALARM B outputs.

To enable RTC_ALARM, configure the OSEL[1:0] bits in the RTC_CR register. These bits select either the alarm A, alarm B, or wake-up outputs.

21.3.3 GPIOs Controlled by RTC

The GPIOs included in the Battery Backup Domain (VBAT) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT and RTC_TS are mapped on the same pin (PE9), while RTC_REFIN is mapped on the pin (PB11).

Table 21-2 PE9 Configuration

PE9 Pin Function		OSEL[1:0] (ALARM Output Enable)	CALBOE (Calibration Output Enable)	ALRM_TYPE	ALRM_PU	TSEN (RTC_TS Input Enable)
RTC_ALARM output Push-Pull		01: Alarm A output enabled 10: Alarm B output enabled 11: Wakeup output enabled	Don't care	0	0	Don't care
RTC_ALARM output Open-Drain	No pull	01: Alarm A output enabled 10: Alarm B output enabled 11: Wakeup output enabled	Don't care	1	0	Don't care
	Internal pull up	01: Alarm A output enabled 10: Alarm B output enabled 11: Wakeup output enabled	Don't care	1	1	Don't care
RTC_CALIB output		00: Output disabled	1	Don't care	Don't care	Don't care
RTC_TS input floating		00: Output disabled	0	Don't care	Don't care	1
Standard GPIO		00: Output disabled	0	Don't care	Don't care	0

21.3.4 Clock and Prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the ELS clock, the ILS oscillator clock, and the EHS clock. For additional details regarding RTC clock source configuration, please refer to Reset and Clock Control (RCC).

A programmable prescaler stage generates a 1 Hz clock, which is used to update the calendar. To reduce power usage, the prescaler is divided into two programmable prescalers (refer to Figure 21-1):

- A 7-bit asynchronous prescaler configured via the PSCDIVA bits of the RTC_PSCR register.
- A 15-bit synchronous prescaler configured via the PSCDIVS bits of the RTC_PSCR register.

NOTE: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to reduce power consumption.

The asynchronous prescaler division factor is set to 127, and the synchronous division factor to 255, to obtain an internal clock frequency of 1 Hz (ck_spre) with an ELS frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}. This corresponds to a maximum input frequency of around 4 MHz.

The calculation for f_{ck_apre} is as follows:

$$f_{ck\_apre} = \frac{fRTCCLK}{PSCDIVA + 1} \quad (7)$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds down-counter. When it reaches 0, RTC_SSR is reloaded with the content of PSCDIVS.

The calculation for f_{ck_spre} is as follows:

$$f_{ck\_spre} = \frac{fRTCCLK}{(PSCDIVS + 1) \times (PSCDIVA + 1)} \quad (8)$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. For shorter timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (refer to Periodic Auto-wakeup for more information).

21.3.5 Real-Time Clock and Calendar

The RTC calendar time and date registers can be accessed via shadow registers, which are synchronized with PCLK (APB clock). Alternatively, they can be accessed directly to avoid the need to wait for the synchronization duration.

- RTC_SSR for the sub seconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (refer to RTC_SHIFTR).

NOTE: The copy is not performed in stop and standby mode. When exiting these modes, the shadow registers are updated after up to 2 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSSHDW control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSSHDW = 0 mode, the frequency of the APB clock must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

21.3.6 Programmable Alarms

The RTC module provides two programmable alarms: Alarm A and Alarm B. The following description details alarm A, but the same applies to alarm B.

The programmable alarm function is enabled by settting the ALRMAEN bit in the RTC_CR register.

The ALRMAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the SECMASK, MINMASK, HOURMASK and DATEMASK bits of the RTC_ALRMAR register, and through the MASKSS bits of the RTC_ALRMASSR register.

To enable the alarm interrupt, use the ALARMIEN bit in the RTC_CR register.

NOTE: If you've selected the seconds field (SECMASK bit has been reset in RTC_ALRMAR), the synchronous prescaler should be configured accordingly. The RTC_PSCR register must be set to a division factor of at least 3 to ensure accurate operation.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the RTC_ALARM output. RTC_ALARM output polarity can be configured through bit POL the RTC_CR register.

If the OSEL[1:0] bits in the RTC_CR register are enabled, both alarm A and alarm B can be routed to the RTC_ALARM output. The polarity of the RTC_ALARM output can be configured by setting the POL bit in the RTC_CR register.

21.3.7 Periodic Auto-Wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup function can be enabled by setting the WUTEN bit in the RTC_CR register.

The wakeup timer clock input can be either the RTC clock (RTCCLK) divided by 2, 4, 8, or 16, or the ck_spre (usually a 1 Hz internal clock).

When using the RTC clock as the wakeup timer clock input and the RTCCLK is 32.768 KHz, the wakeup interrupt period can be configured from 122μs to 32s, with a resolution down to 61μs.

When using ck_spre as the wakeup timer clock input and the frequency of ck_spre is 1 Hz, the wakeup time can be set from 1s to around 36 hours with a one-second resolution. This large programmable time range is divided into two parts:

- From 1s to 18h when WUTCKSELI[2:0] = 10x
- From around 18h to 36h when WUTCKSEL[2:0] = 11x. In this case, 2^{16} is added to the 16-bit counter current value.

After the initialization sequence is complete, the timer starts counting down. The down-counting remains active in low-power modes when the wakeup function is enabled. When the timer reaches 0, the WUTF flag is set in the RTC_SR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value). The WUTF flag must be cleared by software.

The periodic wakeup interrupt can be enabled by setting the WUTIEN bit in the RTC_CR register. It can be used to exit the device from low-power modes.

The periodic wakeup flag can be routed to the RTC_ALARM output if bits OSEL[1:0] in the RTC_CR register have been enabled. The polarity of the RTC_ALARM output can be configured through the POL bit in the RTC_CR register.

It is important to note that the wakeup timer is not affected by system reset or low-power modes such as Sleep, Stop, and Standby.

21.3.8 RTC Initialization and Configuration

21.3.8.1 RTC Register Access

The RTC registers are 32-bit registers. When accessing the RTC registers through the APB interface, there are typically 2 wait-states introduced, except for read accesses to the calendar shadow registers when the BYPSSHDW bit is set to 0.

21.3.8.2 RTC Register Write Protection

After the system reset, the RTC registers are protected against parasitic write access by the DBP bit in the PMU_CR00 register. To enable write access to the RTC registers, the DBP bit must be set.

Following the backup domain reset, certain RTC registers are write-protected.

Writing to the protected RTC registers is enabled by writing a key into the RTC_WPR.

To unlock the write protection on the protected RTC registers, the following steps should be followed:

1. Write the value 0xCA into the RTC_WPR register.

2. Write the value 0x53 into the RTC_WPR register.

NOTE: If an incorrect key is written, the write protection will be reactivated. The protection mechanism is not affected by system reset.

21.3.8.3 Calendar Initialization and Configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, follow the sequence below:

1. Set the INIT bit to 1 in the RTC_ICSR register to enter initialization mode.

   In this mode, the calendar counter is stopped and its value can be updated.
2. Poll the INITF bit in the RTC_ICSR register.

   When INITF is set to 1, it indicates that the initialization phase mode has been entered. It takes approximately 2 RTCCLK clock cycles due to clock synchronization.
3. Program both the prescaler factors in RTC_PSCR register to generate a 1 Hz clock for the calendar counter.
4. Load the initial time and date values into the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Clear the INIT bit to exit the initialization mode.

   The actual calendar counter value will be automatically loaded and counting will restart after 4 RTCCLK clock cycles.

Once the initialization sequence is complete, the calendar starts counting.

NOTE: After a system reset, the application can read the INITSTS flag in the RTC_ICSR register to check if the calendar has been initialized or not.

If the INITSTS flag is 0, it indicates tthat the calendar has not been initialized since the year field is set at its backup domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

21.3.8.4 Daylight Saving Time

The management of daylight saving time in the system is performed through three bits in the RTC_CR register: SUB1H, ADD1H, and BKP.

The SUB1H and ADD1H bits enable the software to subtract or add one hour to the calendar in a single operation, eliminating the need for the initialization procedure.

Additionally, the software can utilize the BKP bit to store and retain this operation.

21.3.8.5 Alarm Programming

A similar procedure must be followed to program or update the programmable alarms. The following procedure is provided for alarm A, but it can be applied in the same way for alarm B

1. Clear the ALRMAEN bit in RTC_CR register to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).
3. Set the ALRMAEN bit in RTC_CR register to enable alarm A again.

NOTE: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

21.3.8.6 Wakeup Timer Programming

To configure or change the wakeup timer auto-reload value (WUTRLDVAL[15:0] in RTC_WUTR register), follow these steps:

1. Clear the WUTEN bit in RTC_CR register to disable the wakeup timer.
2. Poll the WUTWF bit until it is set in RTC_ICSR register to make sure the access to wakeup auto-reload counter and to WUTCKSEL[2:0] bits is allowed.

   NOTE: This step should be skipped in calendar initialization mode. It takes around 2 RTCCLK clock cycles due to clock synchronization.

3. Program the wakeup auto-reload value (WUTRLDVAL[15:0] bits in RTC_WUTR register) and the wakeup clock selection (WUTCKSEL[2:0] bits in RTC_CR register).
4. Set the WUTEN bit in RTC_CR register to enable the timer again.

   The wakeup timer will restart down-counting. The WUTWF bit is cleared up to 2 RTCCLK clocks cycles after WUTEN is cleared due to clock synchronization.

21.3.9 Calendar Reading

21.3.9.1 When BYPSSHDW Bit Is Cleared in RTC_CR Register

To read the RTC calendar registers (RTC_SSR, RTC_TR, and RTC_DR) properly, the APB1 clock frequency (f_{PCLK}) must be at least seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software should read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise, a third read access must be done. In any case, the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in the RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR, and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the three values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. If the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK period, RSF must be cleared by the software after the first calendar read. Then, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR, and RTC_DR registers again.

After a system reset or initialization, the software must wait for the RSF bit to be set before reading the RTC_SSR, RTC_TR, and RTC_DR registers. This is because a system reset or initialization resets the shadow registers to their default values.

Similarly, after synchronization or waking up from a low-power mode (stop or standby), the software must clear the RSF bit and wait for it to be set again before reading the RTC_SSR, RTC_TR, and RTC_DR registers. This ensures that the calendar registers are properly synchronized and consistent.

21.3.9.2 When BYPSSHDW Bit Is Set in RTC_CR Register (Bypass Shadow Registers)

Reading the calendar registers gives the values from the calendar counters directly, eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (such as stop or standby), since the shadow registers are not updated during these modes.

When the BYPSSHDW bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the

data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

NOTE: While BYPSSHDW = 1, instructions which read the calendar registers require one extra APB cycle to complete.

21.3.10 RTC Resetting

The calendar shadow registers (RTC_SSR, RTC_TR, and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a backup domain reset and are not affected by a system reset:

- RTC current calendar registers
- RTC control register (RTC_CR)
- RTC Prescaler register (RTC_PSCR)
- RTC calibration register (RTC_CALR)
- RTC shift register (RTC_SHIFTR)
- RTC timestamp registers (RTC_TSSSR, RTC_TSTR, and RTC_TSDR)
- Wakeup timer register (RTC_WUTR)
- Alarm A and Alarm B registers (RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR)

Additionally, when the RTC is clocked by ELS, it continues running under system reset if the reset source is different from the backup domain reset (refer to Reset and Clock Control (RCC) for details about RTC clock sources). When a backup domain reset occurs, the RTC is stopped, and all the RTC registers are set to their reset values.

21.3.11 RTC Synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by shifting its clock by a fraction of a second using the RTC_SHIFTR register.

RTC_SSR contains the value of the synchronous prescaler counter, enabling the precise calculation of the current time maintained by the RTC with a resolution of 1/(PSCDIVS + 1) seconds. Consequently, the resolution can be enhanced by increasing the value of the synchronous prescaler (PSCDIVS[14:0]). The highest achievable resolution (30.52 µs with a 32768 Hz clock) is attained when PSCDIVS is set to 0x7FFF.

However, increasing PSCDIVS requires decreasing PSCDIVA to keep the synchronous prescaler output at 1 Hz. This results in an increased frequency of the asynchronous prescaler output, potentially leading to higher RTC dynamic consumption.

The RTC can be precisely adjusted using the RTC_SHIFTR. By writing to RTC_SHIFTR, it is possible to shift the clock either delay or advance by up to one second, with a resolution of 1/(PSCDIVS + 1) seconds. The shift operation involves adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0], which causes a delay in the clock. If the ADD1S bit is also set, the clock will be advanced by one second while simultaneously subtracting a fraction of a second.

NOTE: Before initiating a shift operation, the user shall check that the SS[15] bit is 0 to ensure no overflow will occur.

Once a write to the RTC_SHIFTR register initiates a shift operation, the SHIFTF flag is set by hardware to indicate a pending shift operation. This bit is cleared by hardware when the shift operation is completed.

NOTE: This synchronization feature is not compatible with the reference clock detection feature. Firmware must not write to the RTC_SHIFTR register when the REFCKON bit is set to 1.

21.3.12 RTC Reference Clock Detection

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 KHz ELS clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the ELS, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the ELS clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 KHz quartz. The detection is performed during a time window around each of the calendar updates (every 1s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the ELS clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PSCDIVA and PSCDIVS must be set to their default values:

- PSCDIVA = 0x007F
- PSCDIVS = 0x00FF

NOTE: RTC_REFIN clock detection is not available in Standby mode.

21.3.13 RTC Smooth Digital Calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the calibration cycle (the calibration cycle period is 32 seconds in default):

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked.
- Setting CALM[2] to 1 causes four additional cycles to be masked.
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times \left[1 + (CALP \times 512 - CALM)/\left(2^{20} + CALM - CALP \times 512\right)\right] \quad (9)$$

21.3.13.1 Calibration When PSCDIVA < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PSCDIVA bits in RTC_PSCR register) is less than 3. If CALP was already set to 1 and PSCDIVA bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PSCDIVA less than 3, the synchronous prescaler value (PSCDIVS) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PSCDIVA equals 1 (division factor of 2), PSCDIVS should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PSCDIVA equals 0, PSCDIVS should be set to 32759 rather than 32767 (8 less).

If PSCDIVS is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times \left[1 + (256 - CALM)/\left(2^{20} + CALM - 256\right)\right] \quad (10)$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

21.3.13.2 RTC Calibration Verifying

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW 16 bit of the RTC_CALR register can be set to 1 to force a 16-second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when calw16 is set to 1.
- CALW 8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when calw8 is set to 1.

21.3.13.3 Recalibration On-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALBF(recalibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. The INITF is then automatically set to 1.
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

21.3.14 Timestamp Function

Timestamp function can be triggered by the external I/O or the power supply switch to the VBAT.

Timestamp is enabled by setting the TSEN or ITSEN bits of RTC_CR register to 1.

When TSEN is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When ITSEN is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the VBAT supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIEN bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

NOTE: TSF is set 2 ck_apre cycles after the timestamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as 1 while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

CAUTION

If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, both the TSF and TSOVF bits will be set. To prevent masking a timestamp event that occurs at the same moment, the application must not write a value of 0 into the TSF bit unless it has already read it as 1.

21.3.15 Calibration Clock Output

When the CALBOE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the CALBOSEL bit in the RTC_CR register is reset and PSCDIVA = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 KHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When CALBOSEL is set and “PSCDIVS+1” is a non-zero multiple of 256 (i.e: PSCDIVS[7:0] = 0xFF), the CALIB frequency is f_{RTCCLK} / (256 * (PSCDIVA +1)). This corresponds to a calibration output at 1 Hz for prescaler default values (PSCDIVA = 0x7F, PSCDIVS = 0xFF), with an RTCCLK frequency at 32.768 KHz.

NOTE: When the CALIB output is selected, the RTC_OUT pin is automatically configured.

When CALBOSEL is cleared, the CALIB output is the output of the 6^{th} stage of the asynchronous prescaler.

When CALBOSEL is set, the CALIB output is the output of the 8^{th} stage of the synchronous prescaler.

21.3.16 Alarm Output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output RTC_ALARM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

If OSEL ≠ 00, the signal on RTC_ALARM provides both alarm A and alarm B, or wakeup flag.

The polarity of the RTC_ALARM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

21.3.16.1 RTC_ALARM Output

The RTC_ALARM pin can be configured in output open drain or output push-pull using the control bit ALRMTYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to ALRMPU in the RTC_CR register.

NOTE: Once the RTC_ALARM output is enabled, it has priority over CALIB on RTC_OUT.

When RTC_ALARM output is selected, the RTC_OUT pin is automatically configured. In case the RTC_ALARM is configured open-drain in the RTC, the RTC_OUT GPIO must be configured as input.

21.4 Low-Power Modes

Table 21-3 Effect of Low-Power Modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is ELS or ILS. RTC interrupts cause the device to exit the stop mode.
Standby	The RTC remains active when the RTC clock source is ELS or ILS. RTC interrupts cause the device to exit the standby mode.
Shutdown	The RTC remains active when the RTC clock source is ELS. RTC interrupts cause the device to exit the shutdown mode.

Table 21-4 summarizes the RTC pins and functions capability in all modes.

Table 21-4 RTC Pins Functionality over Modes

Functions	Functional in All Low-Power Modes except Standby and Shutdown Modes	Functional in Standby and Shutdown Mode	Functional in VBAT Mode
RTC_TS	Yes	No	No
RTC_REFIN	Yes	No	No
RTC_OUT	Yes	No	No

21.5 Interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

Table 21-5 Interrupt Requests

Interrupt Acronym	Interrupt Event	Event Flag (1)	Enable Control Bit (2)	Interrupt Clear Method	Exit from Sleep Mode	Exit from Stop and Standby Mode	Exit from Shutdown Mode
RTC	Alarm A	ALRMA_F	ALRMAIEN	write 1 in ALRMAFCLR	Yes	Yes (3)	Yes (4)
	Alarm B	ALRMB_F	ALRMBIEN	write 1 in ALRMBFCLR	Yes	Yes (3)	Yes (4)
	Timestamp	TSF	TSIEN	write 1 in TSFCLR	Yes	Yes (3)	Yes (4)
	Wakeup timer interrupt	WUTF	WUTIEN	write 1 in WUTFCLR	Yes	Yes (3)	Yes (4)

(1) The event flags are in the RTC_SR register.

(2) The interrupt enable control bits are in the RTC_CR register.

(3) Wakeup from stop and standby modes is possible only when the RTC clock source is ELS or ILS.

(4) Wakeup from shutdown modes is possible only when the RTC clock source is ELS.

21.6 Registers

21.6.1 RTC Registers

21.6.1.1 Register Address Map

Offset	Register Name	Register Description
0x0000	RTC_TR	Time register Hour & Minute & Second configuration
0x0004	RTC_DR	Date register Year & Month & Week & Date configuration
0x0008	RTC_SSR	Sub-second readback
0x000c	RTC_ICSR	Initialization control and status register This register is write-protected.
0x0010	RTC_PSCR	Prescaler register This register must be written in initialization mode only.
0x0014	RTC_WUTR	Wakeup timer register This register can be written only when WUTWF is set to 1 in RTC_ICSR.
0x0018	RTC_CR	Control register This register is write-protected.
0x001c	RTC_WPR	Write protection register
0x0020	RTC_CALR	Calibration register This register is write-protected.
0x0024	RTC_SHIFTR	Shift control register This register is write-protected.
0x0028	RTC_TSTR	Timestamp time register

Offset	Register Name	Register Description
		The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.
0x002c	RTC_TSDR	Timestamp date register The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.
0x0030	RTC_TSSSR	Timestamp sub second register The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.
0x0034	RTC_ALRMAR	Alarm A register This register can be written only when ALRAWF is set to 1 in RTC_ICSR or in initialization mode.
0x0038	RTC_ALRMASSR	Alarm A sub second register This register can be written only when ALRAWF is set to 1 in RTC_ICSR or in initialization mode.
0x003c	RTC_ALRMBR	Alarm B register This register can be written only when ALRBWF is set to 1 in RTC_ICSR or in initialization mode.
0x0040	RTC_ALRMBSSR	Alarm B sub second register This register can be written only when ALRBWF is set to 1 in RTC_ICSR or in initialization mode.
0x0044	RTC_SR	Status register Backup domain reset value: 0x0000 0000
0x0048	RTC_INTSR	Interrupt status register Backup domain reset value: 0x0000 0000
0x004c	RTC_SCR	Status clear register

Offset	Register Name	Register Description
		Backup domain reset value: 0x0000 0000

21.6.1.2 Register Field Details

21.6.1.2.1 RTC_TR

0x0000			Time register											RTC_TR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved									PM	HOURT		HOURU			
Type	RO									RW	RW		RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	MINT			MINU				Reserved	SECT			SECU			
Type	RO	RW			RW				RO	RW			RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-6 Time Register Description

Field	Name	Description
31:23	Reserved	Reserved
22	PM	AM/PM notation 0: AM or 24-hour format 1: PM
21:20	HOURT	Hour tens in BCD format
19:16	HOURU	Hour units in BCD format
15	Reserved	Reserved
14:12	MINT	Minute tens in BCD format
11:8	MINU	Minute units in BCD format

Field	Name	Description
7	Reserved	Reserved
6:4	SECT	Second tens in BCD format
3:0	SECU	Second units in BCD format

21.6.1.2.2 RTC_DR

0x0004			Date register											RTC_DR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								YEART				YEARU			
Type	RO								RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	WEEKU			MONTHT	MONTHU				Reserved		DATET		DATEU			
Type	RW			RW	RW				RO		RW		RW			
Reset	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1

Table 21-7 Date Register Description

Field	Name	Description
31:24	Reserved	Reserved
23:20	YEART	Year units in BCD format
19:16	YEARU	Year units in BCD format
15:13	WEEKU	WeekDay units in BCD format 000: Forbidden 001: Monday ... 111: Sunday
12	MONTHT	Month tens in BCD format
11:8	MONTHU	Month units in BCD format
7:6	Reserved	Reserved

Field	Name	Description
5:4	DATET	Date tens in BCD format
3:0	DATEU	Date units in BCD format

21.6.1.2.3 RTC_SSR

0x0008			Sub-Second readback											RTC_SSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SS															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-8 Sub-Second Readback Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	SS	Sub seconds value SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below: Second fraction = (PSCDIVS - SS) / (PSCDIVS + 1) **NOTE:** SS can be larger than PSCDIVS only after a shift operation. In that case, the correct time/date is one second less than indicated by RTC_TR/RTC_DR.

21.6.1.2.4 RTC_ICSR

0x000c			Initialization control and status register											RTC_ICSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															RECALBF
Type	RO															RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								INIT	INITF	RSF	INITSTS	SHIFTF	WUTWF	ALRMBWF	ALRMAWF
Type	RO								RW	RO	RC_W0	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Table 21-9 Initialization Control and Status Register Description

Field	Name	Description
31:17	Reserved	Reserved
16	RECALBF	Recalibration pending flag The RECALBF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0.
15:8	Reserved	Reserved
7	INIT	Initialization mode 0: Free running mode 1: Initialization mode used to program time and date register (RTC_TR and RTC_DR) and prescaler register (RTC_PSCR). Counters are stopped and start counting from the new value when INIT is reset.

Field	Name	Description
6	INITF	Initialization flag When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated. 0: Calendar registers update is not allowed 1: Calendar registers update is allowed
5	RSF	Registers synchronization flag This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR, and RTC_DR). This bit is cleared by hardware in the initialization mode while a shift operation is pending (SHIFTF = 1) or when in bypass shadow register mode (BYPSSHDW = 1). This bit can also be cleared by software. It is cleared either by software or by hardware in initialization mode. 0: Calendar shadow registers not yet synchronized 1: Calendar shadow registers synchronized
4	INITSTS	Initialization status flag This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state). 0: Calendar has not been initialized 1: Calendar has been initialized
3	SHIFTF	Shift operation pending This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHIFTF bit has no effect. 0: No shift operation is pending 1: A shift operation is pending
2	WUTWF	Wakeup timer write flag This bit is set by hardware when the WUTRLDVAL value can be changed after the WUTEN bit has been set to 0 in RTC_CR.

Field	Name	Description
		It is cleared by hardware in initialization mode. 0: Wakeup timer configuration update not allowed except in initialization mode 1: Wakeup timer configuration update allowed
1	ALRMBWF	Alarm B write flag This bit is set by hardware when alarm B values can be changed after the ALRMBEN bit has been set to 0 in RTC_CR. It is cleared by hardware in initialization mode. 0: Alarm B update not allowed 1: Alarm B update allowed
0	ALRMAWF	Alarm A write flag This bit is set by hardware when alarm A values can be changed after the ALRMAEN bit has been set to 0 in RTC_CR. It is cleared by hardware in initialization mode. 0: Alarm A update not allowed 1: Alarm A update allowed

21.6.1.2.5 RTC_PSCR

0x0010			Prescaler register											RTC_PSCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved									PSCDIVA						
Type	RO									RW						
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	PSCDIVS														
Type	RO	RW														
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Table 21-10 Prescaler Register Description

Field	Name	Description
31:23	Reserved	Reserved
22:16	PSCDIVA	Asynchronous prescaler factor This is the asynchronous division factor: ck_apre frequency = RTCCLK frequency / (PSCDIVA + 1)
15:15	Reserved	Reserved
14:0	PSCDIVS	Synchronous prescaler factor This is the synchronous division factor: ck_spre frequency = ck_apre frequency / (PSCDIVS + 1)

21.6.1.2.6 RTC_WUTR

0x0014			Wakeup timer register											RTC_WUTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	WUTRLDVAL															
Type	RW															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 21-11 Wakeup Timer Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	WUTRLDVAL	Wakeup auto-reload value bits When the wakeup timer is enabled (WUTEN set to 1), the WUTF flag is set every (WUTRLDVAL[15:0] + 1) CKWUT cycles. The CKWUT period is selected through WUTCKSEL[2:0] bits of the RTC_CR register. When WUTCKSEL[2] = 1, the wakeup timer becomes 17-bit and WUTCKSEL[1] effectively becomes WUTRLDVAL[16] the most-significant bit to be reloaded into the timer. The first assertion of WUTF occurs between WUTRLDVAL and (WUTRLDVAL + 1) CKWUT cycles after WUTEN is set. Setting WUTRLDVAL[15:0] to 0x0000 with WUTCKSEL[2:0] = 011 (RTCCLK / 2) is forbidden.

21.6.1.2.7 RTC_CR

0x0018			Control register											RTC_CR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved	ALRMTYPE	ALRMPU	Reserved				ITSEN	CALBOE	OSEL		POL	CALBOSEL	BKP	SUB1H	ADD1H
Type	RO	RW	RW	RO				RW	RW	RW		RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TSIEN	WUTIEN	ALRMBIEN	ALRMAIEN	TSEN	WUTEN	ALRMBEN	ALRMAEN	Reserved	FMT	BYPSSHDW	REFCKON	TSEDGE	WUTCKSEL		
Type	RW	RW	RW	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-12 Control Register Description

Field	Name	Description
31	Reserved	Reserved
30	ALRMTYPE	ALRM output type 0: ALRM is push-pull output 1: ALRM is open-drain output
29	ALRMPU	ALRM pull-up enable 0: No pull-up is applied on ALRM output 1: A pull-up is applied on ALRM output
28:25	Reserved	Reserved
24	ITSEN	Timestamp on internal event enable 0: Internal event timestamp disabled

Field	Name	Description
		1: Internal event timestamp enabled
23	CALBOE	Calibration output enable This bit enables the CALIB output 0: Calibration output disabled 1: Calibration output enabled
22:21	OSEL	Output selection These bits are used to select the flag to be routed to ALRM output. 00: Output disabled 01: Alarm A output enabled 10: Alarm B output enabled 11: Wakeup output enabled
20	POL	Output polarity This bit is used to configure the polarity of ALRM output. 0: The pin is high when ALRMAF/ALRMBF/WUTF is asserted (depending on OSEL[1:0]). 1: The pin is low when ALRMAF/ALRMBF/WUTF is asserted (depending on OSEL[1:0]).
19	CALBOSEL	Calibration output selection When CALBOE = 1, this bit selects which signal is output on CALIB. 0: Calibration output is 512 Hz 1: Calibration output is 1 Hz These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PSCDIVA = 127 and PSCDIVS = 255).
18	BKP	Backup The user can write this bit to memorize whether the daylight saving time change has been performed or not.
17	SUB1H	Subtract 1 hour (winter time change) When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Field	Name	Description
		Setting this bit has no effect when the current hour is 0. 0: No effect 1: Subtracts 1 hour to the current time. This can be used for wintertime change.
16	ADD1H	Add 1 hour (summer time change) When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0. 0: No effect 1: Adds 1 hour to the current time. This can be used for summer time change.
15	TSIEN	Time-stamp interrupt enable 0: Timestamp interrupt disable 1: Timestamp interrupt enable
14	WUTIEN	Wakeup timer interrupt enable 0: Wakeup timer interrupt disabled 1: Wakeup timer interrupt enabled
13	ALRMBIEN	Alarm B interrupt enable 0: Alarm B interrupt disable 1: Alarm B interrupt enable
12	ALRMAIEN	Alarm A interrupt enable 0: Alarm A interrupt disabled 1: Alarm A interrupt enabled
11	TSEN	Timestamp enable 0: Timestamp disable 1: Timestamp enable
10	WUTEN	Wakeup timer enable 0: Wakeup timer disabled 1: Wakeup timer enabled

Field	Name	Description
		NOTE: When the wakeup timer is disabled, wait for WUTWF = 1 before enabling it again.
9	ALRMBEN	Alarm B enable 0: Alarm B disabled 1: Alarm B enabled
8	ALRMAEN	Alarm A enable 0: Alarm A disabled 1: Alarm A enabled
7	Reserved	Reserved
6	FMT	Hour format 0: 24 hour/day format 1: AM/PM hour format
5	BYPSSHDW	Bypass the shadow registers 0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles. 1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters. **NOTE:** If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSSHDW must be set to 1.
4	REFCKON	RTC_REFIN reference clock detection enable (50 or 60 Hz) 0: RTC_REFIN detection disabled 1: RTC_REFIN detection enabled **NOTE:** The value of PSCDIVS should be 0x00FF.

Field	Name	Description
3	TSEDGE	Timestamp event active edge 0: RTC_TS input rising edge generates a timestamp event 1: RTC_TS input falling edge generates a timestamp event TSEN must be reset when TSEDGE is changed to avoid unwanted TSF setting.
2:0	WUTCKSEL	ck_wut wakeup clock selection 000: RTC/16 clock is selected 001: RTC/8 clock is selected 010: RTC/4 clock is selected 011: RTC/2 clock is selected 10x: ck_spre (usually 1 Hz) clock is selected 11x: ck_spre (usually 1 Hz) clock is selected and 2**16 is added to the wut_rld_val counter value

21.6.1.2.8 RTC_WPR

0x001c														RTC_WPR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								KEY							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-13 Control Register Description

Field	Name	Description
31:8	Reserved	Reserved
7:0	KEY	Write protection key This byte is written by software. Reading this byte always returns 0x00.

21.6.1.2.9 RTC_CALR

0x0020			Calibration register											RTC_CALR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CALP	CALW8	CALW16	Reserved				CALM								
Type	RW	RW	RW	RO				RW								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-14 Calibration Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	CALP	Increase frequency of RTC by 488.5 ppm 0: No RTCCLK pulses are added. 1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm). This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 x CALP) - CALM.
14	CALW8	Use an 8-second calibration cycle period When CALW8 is set to 1, the 8-second calibration cycle period is selected. **NOTE:** CALM[1:0] are stuck at 00 when CALW8 = 1.

Field	Name	Description
13	CALW16	Use a 16-second calibration cycle period When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1. **NOTE:** CALM[0] is stuck at 0 when CALW16 = 1.
12:9	Reserved	Reserved
8:0	CALM	Calibration minus The frequency of the calendar is reduced by masking CALM out of 2^20 RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm. To increase the frequency of the calendar, this feature should be used in conjunction with CALP.

21.6.1.2.10 RTC_SHIFTR

0x0024			Shift control register											RTC_SHIFTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	ADD1S	Reserved														
Type	RW	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	SUBFS														
Type	RO	RW														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-15 Shift Control Register Description

Field	Name	Description
31	ADD1S	Add one second 0: No effect 1: Add one second to the clock/calendar This bit is written only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHIFTF = 1, in RTC_ICSR). This function is intended to be used with SUBFS (see description below) to effectively add a fraction of a second to the clock in an atomic operation.
30:15	Reserved	Reserved
14:0	SUBFS	Subtract a fraction of a second These bits are write-only and are always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHIFTF = 1, in RTC_ICSR).

Field	Name	Description
		The value written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by: Delay (seconds) = SUBFS / (PREDIVS + 1) A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by: Advance (seconds) = (1 - (SUBFS / (PREDIVS + 1))). **NOTE:** Writing to SUBFS causes RSF to be cleared. The software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

21.6.1.2.11 RTC_TSTR

0x0028			Timestamp time register											RTC_TSTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved									PM	HOURT		HOURU			
Type	RO									RO	RO		RO			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	MINT			MINU				Reserved	SECT			SECU			
Type	RO	RO			RO				RO	RO			RO			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-16 Timestamp Time Register Description

Field	Name	Description
31:23	Reserved	Reserved
22	PM	AM/PM notation 0: AM or 24-hour format 1: PM
21:20	HOURT	Hour tens in BCD format
19:16	HOURU	Hour units in BCD format
15	Reserved	Reserved
14:12	MINT	Minute tens in BCD format
11:8	MINU	Minute units in BCD format
7	Reserved	Reserved

Field	Name	Description
6:4	SECT	Second tens in BCD format
3:0	SECU	Second units in BCD format

21.6.1.2.12 RTC_TSDR

0x002c			Timestamp date register											RTC_TSDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	WEEKU			MONTHT	MONTHU				Reserved		DATET		DATEU			
Type	RO			RO	RO				RO		RO		RO			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-17 Timestamp Date Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:13	WEEKU	WeekDay units in BCD format
12	MONTHT	Month tens in BCD format
11:8	MONTHU	Month units in BCD format
7:6	Reserved	Reserved
5:4	DATET	Date tens in BCD format
3:0	DATEU	Date units in BCD format

21.6.1.2.13 RTC_TSSSR

0x0030			Timestamp sub second rregister											RTC_TSSSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SS															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-18 Timestamp Sub Second Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	SS	Sub second value SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

21.6.1.2.14 RTC_ALRMAR

0x0034			Alarm A register											RTC_ALRMAR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DATEMASK	WEEKSEL	DATET		DATEU				HOURMASK	PM	HOURT		HOURU			
Type	RW	RW	RW		RW				RW	RW	RW		RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MINMASK	MINT			MINU				SECMASK	SECT			SECU			
Type	RW	RW			RW				RW	RW			RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-19 Alarm A Register Description

Field	Name	Description
31	DATEMASK	Alarm A date mask 0: Alarm A set if the date/day match 1: Date/day don't care in alarm A comparison
30	WEEKSEL	Week day selection 0: DATEU[3:0] represents the date units 1: DATEU[3:0] represents the week day. DATEU[1:0] is don't care.
29:28	DATET	Date tens in BCD format
27:24	DATEU	Date units in BCD format
23	HOURMASK	Alarm A hours mask 0: Alarm A set if the hours match 1: Hours don't care in alarm A comparison

Field	Name	Description
22	PM	AM/PM notation 0: AM or 24-hour format 1: PM
21:20	HOURT	Hour tens in BCD format
19:16	HOURU	Hour units in BCD format
15	MINMASK	Alarm A minutes mask 0: Alarm A set if the minutes match 1: Minutes don't care in alarm A comparison
14:12	MINT	Minute tens in BCD format
11:8	MINU	Minute units in BCD format
7	SECMASK	Alarm A seconds mask 0: Alarm A set if the seconds match 1: Seconds don't care in alarm A comparison
6:4	SECT	Second tens in BCD format
3:0	SECU	Second units in BCD format

21.6.1.2.15 RTC_ALRMASSR

0x0038			Alarm A sub second register											RTC_ALRMASSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				MASKSS				Reserved							
Type	RO				RW				RO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	SS														
Type	RO	RW														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-20 Alarm A Sub Second Register Description

Field	Name	Description
31:28	Reserved	Reserved
27:24	MASKSS	Mask the most-significant bits starting at this bit 0000: No comparison on sub-seconds for alarm A. The alarm is set when the seconds unit is incremented (assuming the rest of the fields match). 0001: SS[14:1] don't care in Alarm A comparison. Only SS[0] is compared. 0010: SS[14:2] don't care in Alarm A comparison. Only SS[1:0] are compared. 0011: SS[14:3] don't care in Alarm A comparison. Only SS[2:0] are compared. ... 1100: SS[14:12] don't care in Alarm A comparison. SS[11:0] are compared. 1101: SS[14:13] don't care in Alarm A comparison. SS[12:0] are compared. 1110: SS[14] doesn't care in alarm A comparison. SS[13:0] are compared. 1111: All 15 SS bits are compared and must match to activate the alarm.

Field	Name	Description
		The overflow bits of the synchronous counter (bits 15) are never compared. This bit can be different from 0 only after a shift operation. **NOTE:** The overflow bits of the synchronous counter (bits 15) are never compared. This bit can be different from 0 only after a shift operation.
23:15	Reserved	Reserved
14:0	SS	Sub seconds value This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

21.6.1.2.16 RTC_ALRMBR

0x003c			Alarm B register											RTC_ALRMBR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DATEMASK	WEEKSEL	DATET		DATEU				HOURMASK	PM	HOURT		HOURU			
Type	RW	RW	RW		RW				RW	RW	RW		RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	MINMASK	MINT			MINU				SECMASK	SECT			SECU			
Type	RW	RW			RW				RW	RW			RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-21 Alarm B Register Description

Field	Name	Description
31	DATEMASK	Alarm B date mask 0: Alarm B set if the date/day match 1: Date/day don't care in Alarm B comparison
30	WEEKSEL	Week day selection 0: DATEU[3:0] represents the date units 1: DATEU[3:0] represents the week day. DATEU[1:0] is don't care
29:28	DATET	Date tens in BCD format
27:24	DATEU	Date units in BCD format
23	HOURMASK	Alarm B hours mask 0: Alarm B set if the hours match

Field	Name	Description
		1: Hours don't care in Alarm B comparison
22	PM	AM/PM notation 0: AM or 24-hour format 1: PM
21:20	HOURT	Hour tens in BCD format
19:16	HOURU	Hour units in BCD format
15	MINMASK	Alarm B minutes mask 0: Alarm B set if the minutes match 1: Minutes don't care in Alarm B comparison
14:12	MINT	Minute tens in BCD format
11:8	MINU	Minute units in BCD format
7	SECMASK	Alarm B seconds mask 0: Alarm B set if the seconds match 1: Seconds don't care in Alarm B comparison
6:4	SECT	Second tens in BCD format
3:0	SECU	Second units in BCD format

21.6.1.2.17 RTC_ALRMBSSR

0x0040			Alarm B sub second register											RTC_ALRMBSSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				MASKSS				Reserved							
Type	RO				RW				RO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	SS														
Type	RO	RW														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-22 Alarm B Sub Second Register Description

Field	Name	Description
31:28	Reserved	Reserved
27:24	MASKSS	Mask the most-significant bits starting at this bit 0000: No comparison on sub-seconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming the rest of the fields match). 0001: SS[14:1] don't care in Alarm B comparison. Only SS[0] is compared. 0010: SS[14:2] don't care in Alarm B comparison. Only SS[1:0] are compared. 0011: SS[14:3] don't care in Alarm B comparison. Only SS[2:0] are compared. ... 1100: SS[14:12] don't care in Alarm B comparison. SS[11:0] are compared. 1101: SS[14:13] don't care in Alarm B comparison. SS[12:0] are compared. 1110: SS[14] doesn't care in Alarm B comparison. SS[13:0] are compared. 1111: All 15 SS bits are compared and must match to activate alarm.

Field	Name	Description
		The overflow bits of the synchronous counter (bits 15) are never compared. This bit can be different from 0 only after a shift operation. **NOTE:** The overflow bits of the synchronous counter (bits 15) are never compared. This bit can be different from 0 only after a shift operation.
23:15	Reserved	Reserved
14:0	SS	Sub seconds value This value is compared with the contents of the synchronous prescaler counter to determine if Alarm B is to be activated. Only bits 0 up MASKSS-1 are compared.

21.6.1.2.18 RTC_SR

0x0044			Status register											RTC_SR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										ITSF	TSOVF	TSF	WUTF	ALRMBF	ALRMAF
Type	RO										RO	RW	RW	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-23 Status Register Description

Field	Name	Description
31:6	Reserved	Reserved
5	ITSF	Internal timestamp flag This flag is set by hardware when a timestamp on the internal event occurs.
4	TSOVF	Timestamp overflow flag This flag is set by hardware when a timestamp event occurs while TSF is already set. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.
3	TSF	Timestamp flag This flag is set by hardware when a timestamp event occurs. If ITSF flag is set, TSF must be cleared together with ITSF.
2	WUTF	Wakeup timer flag

Field	Name	Description
		This flag is set by hardware when the wakeup auto-reload counter reaches 0. This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.
1	ALRMBF	Alarm B flag This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).
0	ALRMAF	Alarm A flag This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

21.6.1.2.19 RTC_INTSR

0x0048			Interrupt status register											RTC_INTSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										ITSIF	TSOVF	TSIF	WUTIF	ALRMBIF	ALRMAIF
Type	RO										RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-24 Interrupt Status Register Description

Field	Name	Description
31:6	Reserved	Reserved
5	ITSIF	Internal timestamp interrupt flag This flag is set by hardware when a timestamp on the internal event occurs and timestamp interrupt is raised.
4	TSOVF	Timestamp overflow interrupt flag This flag is set by hardware when a timestamp interrupt occurs while TSIF is already set. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.
3	TSIF	Timestamp interrupt flag This flag is set by hardware when a timestamp interrupt occurs. If ITSF flag is set, TSF must be cleared together with ITSF.
2	WUTIF	Wakeup timer interrupt flag This flag is set by hardware when the wakeup timer interrupt occurs.

Field	Name	Description
		This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.
1	ALRMBIF	Alarm B interrupt flag This flag is set by hardware when the alarm B interrupt occurs.
0	ALRMAIF	Alarm A interrupt flag This flag is set by hardware when the alarm A interrupt occurs.

21.6.1.2.20 RTC_SCR

0x004c			Status Clear Register									RTC_SCR				
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										ITSFCLR	TSOVFCLR	TSFCLR	WUTFCLR	ALRMBFCLR	ALRMAFCLR
Type	RO										RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-25 Status Clear Register Description

Field	Name	Description
31:6	Reserved	Reserved
5	ITSFCLR	Clear internal timestamp flag Writing 1 in this bit clears the ITSF bit in the RTC_SR register.
4	TSOVFCLR	Clear timestamp overflow flag Writing 1 in this bit clears the TSOVF bit in the RTC_SR register. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.
3	TSFCLR	Clear timestamp flag Writing 1 in this bit clears the TSOVF bit in the RTC_SR register. If ITSF flag is set, TSF must be cleared together with ITSF by setting RSF and ITSFCLR.
2	WUTFCLR	Clear wakeup timer flag

Field	Name	Description
		Writing 1 in this bit clears the WUTF bit in the RTC_SR register.
1	ALRMBFCLR	Clear alarm B flag Writing 1 in this bit clears the ALRMBF bit in the RTC_SR register.
0	ALRMAFCLR	Clear alarm A flag Writing 1 in this bit clears the ALRMAF bit in the RTC_SR register.

21.6.2 Backup Registers

21.6.2.1 Register Address Map

Offset	Register Name	Register Description
0x0000	BKP_BKP1R	Backup 1 register
0x0004	BKP_BKP2R	Backup 2 register
0x0008	BKP_BKP3R	Backup 3 register
0x000c	BKP_BKP4R	Backup 4 register
0x0010	BKP_BKP5R	Backup 5 register
0x0014	BKP_BKP6R	Backup 6 register
0x0018	BKP_BKP7R	Backup 7 register
0x001c	BKP_BKP8R	Backup 8 register
0x0020	BKP_BKP9R	Backup 9 register
0x0024	BKP_BKP10R	Backup 10 register
0x0028	BKP_BKP11R	Backup 11 register
0x002c	BKP_BKP12R	Backup 12 register
0x0030	BKP_BKP13R	Backup 13 register
0x0034	BKP_BKP14R	Backup 14 register
0x0038	BKP_BKP15R	Backup 15 register
0x003c	BKP_BKP16R	Backup 16 register

21.6.2.2 Register Field Details

21.6.2.2.1 BKP_BKP1R

0x0000			Backup 1 register											BKP_BKP1R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-26 Backup 1 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection RDP) is disabled.

21.6.2.2.2 BKP_BKP2R

0x0004			Backup 2 register											BKP_BKP2R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-27 Backup 2 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.3 BKP_BKP3R

0x0008			Backup 3 register											BKP_BKP3R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-28 Backup 3 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.4 BKP_BKP4R

0x000c			Backup 4 register											BKP_BKP4R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-29 Backup 4 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.5 BKP_BKP5R

0x0010			Backup 5 register											BKP_BKP5R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-30 Backup 5 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.6 BKP_BKP6R

0x0014			Backup 6 register											BKP_BKP6R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-31 Backup 6 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.7 BKP_BKP7R

0x0018			Backup 7 register											BKP_BKP7R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-32 Backup 7 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.8 BKP_BKP8R

0x001c			Backup 8 register											BKP_BKP8R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-33 Backup 8 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.9 BKP_BKP9R

0x0020			Backup 9 register											BKP_BKP9R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-34 Backup 9 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.10 BKP_BKP10R

0x0024			Backup 10 register											BKP_BKP10R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-35 Backup 10 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.11 BKP_BKP11R

0x0028			Backup 11 register											BKP_BKP11R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-36 Backup 11 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.12 BKP_BKP12R

0x002c			Backup 12 register											BKP_BKP12R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-37 Backup 12 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.13 BKP_BKP13R

0x0030			Backup 13 register											BKP_BKP13R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-38 Backup 13 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.14 BKP_BKP14R

0x0034			Backup 14 register											BKP_BKP14R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-39 Backup 14 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.15 BKP_BKP15R

0x0038			Backup 15 register											BKP_BKP15R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-40 Backup 15 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

21.6.2.2.16 BKP_BKP16R

0x003c			Backup 16 register											BKP_BKP16R		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BKP															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-41 Backup 16 Register Description

Field	Name	Description
31:0	BKP	The application can write or read data to and from these registers. They are powered on by VBAT when VDD is switched off, so they are not reset by a system reset, and their contents remain valid when the device operates in low-power mode. In the default configuration, this register is reset on a tamper detection event. It is forced to reset the value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

Comparator (CMP)

This chapter describes the details of the Comparator (CMP).

Topics: **Page**

22.1 Introduction

The Comparator (CMP) module is an analog voltage comparator with a built-in 6-bit DAC. It compares two analog signals and provides a digital output indicating which is larger.

The CMP supports general comparator functionality for up to 8 channels, including one specified for the built-in DAC, and implements programmable hysteresis, blanking, and trigger break events.

22.2 Features

- Selection from multiple input sources
- Integrate 6-bit DAC for CMP reference voltage
- Programmable hysteresis
- Interrupt capability and wake up the CPU from stop mode
- Interrupt driven measurement system for low-power operation support
- Output redirection to I/Os or to timer inputs for triggering break events for fast PWM shutdowns
- Output blanking for immunity to switching noise

22.3 Functional Description

22.3.1 Block Diagram

Figure 22-1 shows the block diagram of the CMP module.

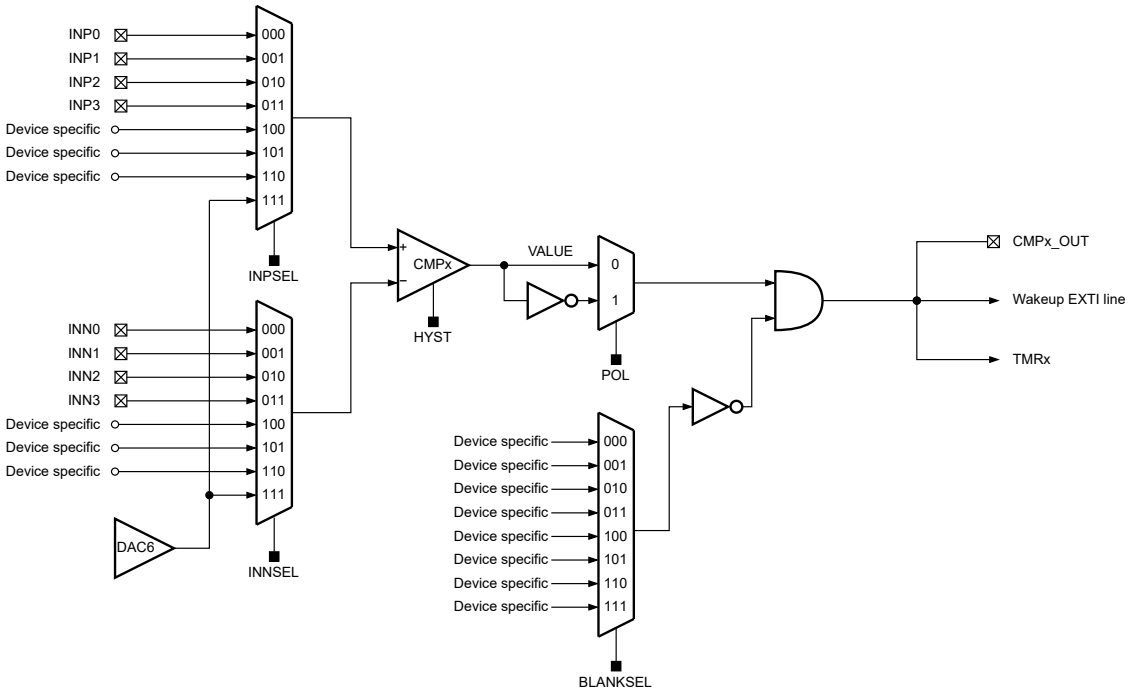


Figure 22-1 CMP Block Diagram

22.3.2 Analog Input Switch

The analog input switches connect or disconnect the two comparator input terminals to associated pins using the INPSEL and INNSEL bits. The comparator terminal inputs can be controlled individually. The INPSEL and INNSEL bits allow:

- Connect external signals to the V+ and V– terminals of the comparator.

- Connect internal signals to the V+ and V– terminals of the comparator.
- Connect an internal reference voltage from the built-in 6-bit DAC to the V+ or V– terminal of the comparator.

Table 22-1 lists the supported configurations for CMP.

Table 22-1 Supported Configurations for CMP

Mode	Value	Description
INPSEL	000	CMPxINP0
	001	CMPxINP1
	010	CMPxINP2
	011	CMPxINP3
	100	Device specific
	101	Device specific
	110	Device specific
	111	Built-in 6-bit DAC
INNSEL	000	CMPxINN0
	001	CMPxINN1
	010	CMPxINN2
	011	CMPxINN3
	100	Device specific
	101	Device specific
	110	Device specific
	111	Built-in 6-bit DAC

22.3.3 CMP Built-in 6-Bit DAC

The CMP integrates a 6-bit DAC as the reference voltage input. The DAC reference voltage can be sourced from V_{DDA}.

The DAC output is enabled or disabled using control bit DAC6BEN. When disabled, the DAC always outputs low. The DAC output can be set in 64 steps from 0 to 63 (full range – 1 LSB of reference voltage).

22.3.4 CMP Hysteresis

To avoid spurious output transitions with noisy input signals, the comparator includes a programmable hysteresis. This hysteresis is non-symmetrical and only responds to the falling edge of the comparator's output. The internal hysteresis function can be disabled to set the amount of hysteresis with external components. This could be beneficial, for instance, when coming out of a low-power mode. Figure 22-2 shows the block diagram of CMP hysteresis.

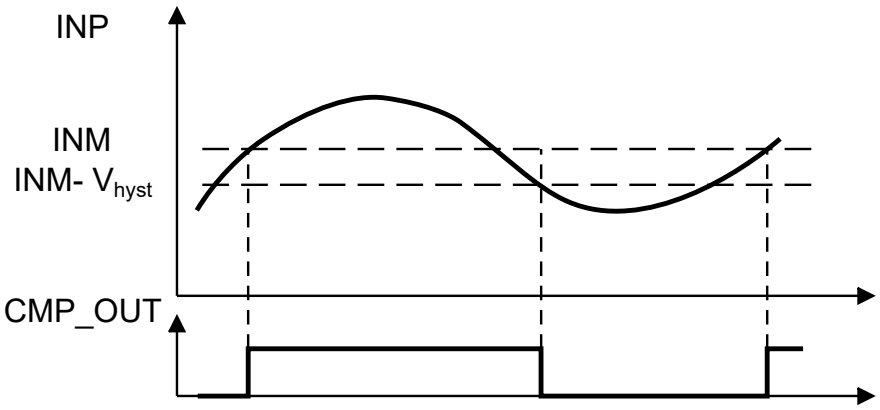


Figure 22-2 CMP Hysteresis Block Diagram

22.3.5 CMP Blanking

The blanking function is used to prevent the current regulation from tripping upon short current spikes at the beginning of PWM period (typically the recovery current in power switch anti-parallel diodes). This goes through setting a dead window defined with a timer output compare signal. The blanking source is selected individually per comparator channel by software through BLANKSEL[2:0] bits. The inverted blanking signal is logical AND-ed with the comparator stage output to produce the comparator channel x output.

Figure 22-3 shows an example of CMP output blanking.

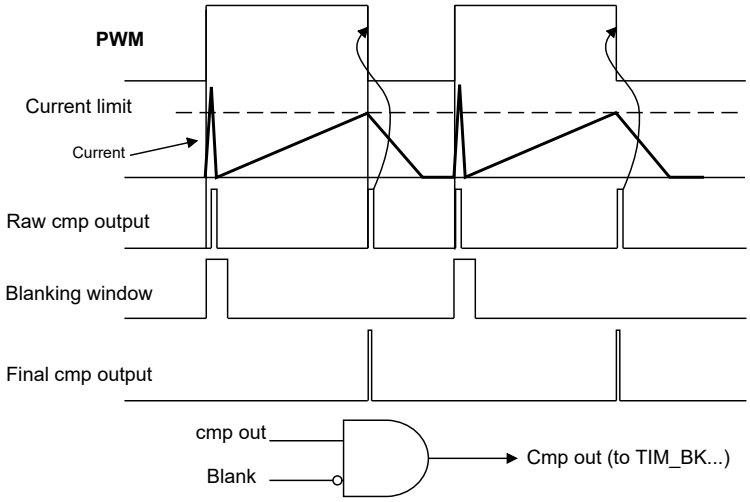


Figure 22-3 CMP Output Blanking Example

22.3.6 CMP Lock Mechanism

In certain applications like over-current, over-voltage, and thermal protection, it is crucial to maintain the comparator programming unchangeable for safety reasons.

This can be accomplished by making the comparator control and status registers write-protected, or read-only. After the programming is finalized, the LOCK bit can be activated, turning the entire register, including the LOCK bit itself, into read-only. The only way to remove the write protection is through a system reset.

22.3.7 CMP Low-Power Modes

The Comparator (CMP) supports low-power sleep and stop modes. The CMP interrupts cause the device to exit these two modes. During standby and shutdown modes, the CMP switches off.

22.4 Interrupts

The Comparator (CMP) outputs are internally connected to the extended interrupts and events controller. Each CMP has its own EXTI line and can generate either interrupts or events.

To enable the CMP interrupt, adhere to the sequence outlined below:

1. Configure and enable the EXTI line corresponding to the CMP output event in interrupt mode and select sensitivity to either the rising edge, falling edge, or both.
2. Configure and enable the NVIC IRQ channel that maps to the corresponding EXTI lines.
3. Enable CMP interrupt events are flagged through the CMP flags bit.

22.5 Registers

22.5.1 Register Address Map

Offset	Register Name	Register Description
0x0000	CMP_CSR	Global Control Register

22.5.2 Register Field Details

22.5.2.1 CMP_CSR

0x0000			Global Control Register											CMP_CSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	LOCK	VALUE	Reserved			DAC6BDATAIN						DAC6BEN	BLANKSEL			BLANKINGEN
Type	RW	RO	RO			RW						RW	RW			RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	HYST			POL	Reserved	INPSEL			Reserved	INNSEL			Reserved			CMPEN
Type	RW			RW	RO	RW			RO	RW			RO			RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-2 Global Control Register Description

Field	Name	Description
31	LOCK	Register lock This bit is set by software and cleared by a hardware system reset. 0: Unlock 1: Lock
30	VALUE	CMP output status This read-only flag reflects the level of the CMP output before the polarity selector and blanking.
29:27	Reserved	Reserved

Field	Name	Description
26:21	DAC6BDATAIN	DAC6B data input
20	DAC6BEN	DAC6B enable 0: Disable 1: Enable
19:17	BLANKSEL	CMP blanking signal select
16	BLANKINGEN	CMP blanking enable 0: Disable 1: Enable
15:13	HYST	CMP hysteresis 000: No hysteresis voltage 001: 5mV 010: 10mV 011: 15mV 100: 20mV 101: 25mV 110: 30mV 111: Reserved
12	POL	CMP polarity This bit controlled by software selects the CMP output polarity: 0: Non-inverted 1: Inverted
11	Reserved	Reserved
10:8	INPSEL	000: INP0 001: INP1 010: INP2 011: INP3

Field	Name	Description
		100: INP4 101: INP5 110: INP6 111: Reserved for DAC6B_OUT
7	Reserved	Reserved
6:4	INNSEL	000: INN0 001: INN1 010: INN2 011: INN3 100: INN4 101: INN5 110: INN6 111: Reserved for DAC6B_OUT
3:1	Reserved	Reserved
0	CMPEN	CMP enable/disable control 0: CMP disable 1: CMP enable

Operational Amplifier (OA)

This chapter describes the details of Operational Amplifier (OA).

Topics: **Page**

23.1 Introduction

The Operational Amplifier (OA) supports the rail-to-rail input and output, with up to 65x gain PGA. It can be used for the signal conditional path.

23.2 Features

- Rail-to-rail input and output voltage range.
- Up to 3 input selections for both negative and positive channels.
- Configurable in General Purpose Amplifier (GP), Programmable Gain Amplifier (PGA), or buffer mode.
- Automatical protection against configuration errors.

23.3 Functional Description

The OA is configurable as GP, PGA or Buffer modes according to applications need. It is also automatically calibrated in end user side if required.

23.3.1 Block Diagram

Figure 23-1 illustrates the overall block diagram of OA.

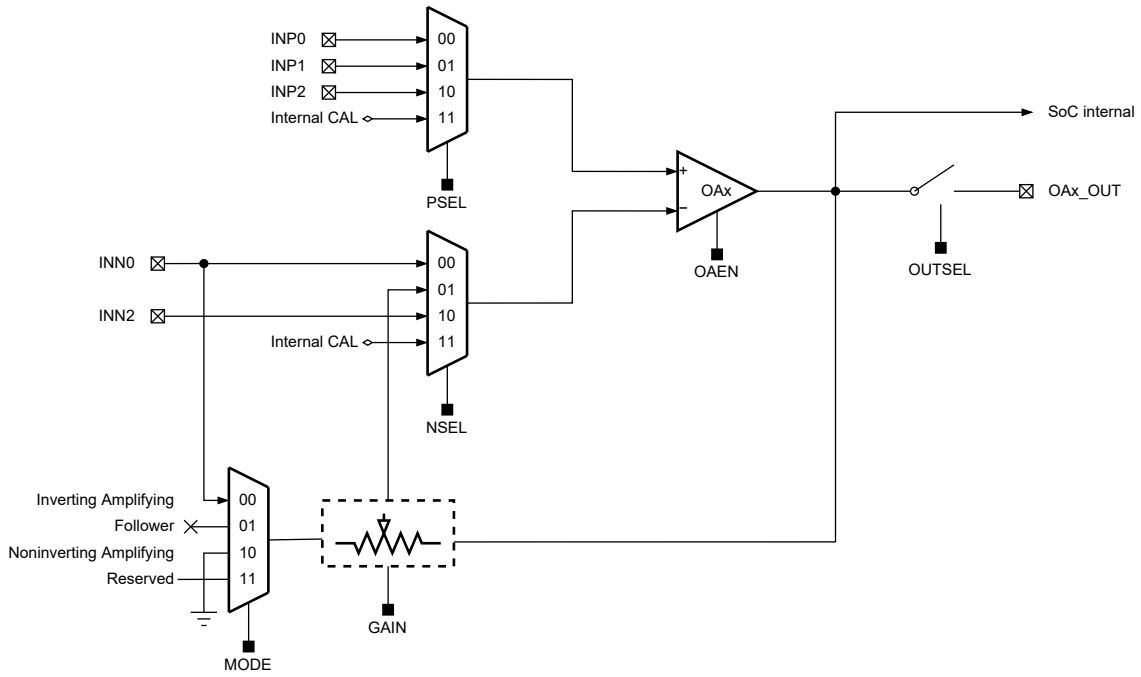


Figure 23-1 OA Block Diagram

23.3.2 OA Modes

Table 23-1 lists the supported configurations for the OA.

Table 23-1 Supported OA Configurations

Mode	MODSEL	PSEL	NSEL	GAIN	Description
GP	XX except 11	00,01,10	00,10	N/A	General-purpose OA by external pin

Mode	MODSEL	PSEL	NSEL	GAIN	Description
Buffer	01	00,01,10	01	N/A	Buffer output of external pin
Inverting PGA	00	00,01,10	01	-1 to -64	Inverting PGA amplifier mode
Non-inverting PGA	10	00,01,10	01	2 to 65	Non-inverting PGA amplifier

23.3.2.1 OA GP Mode

The operational amplifier (OA) is equipped with an internal multiplexer that supports the general purpose (GP) mode. In this mode, OAxINP and OAxINN pins are dedicated to non-inverting and inverting inputs.

Figure 23-2 shows the block diagram of the OA GP mode.

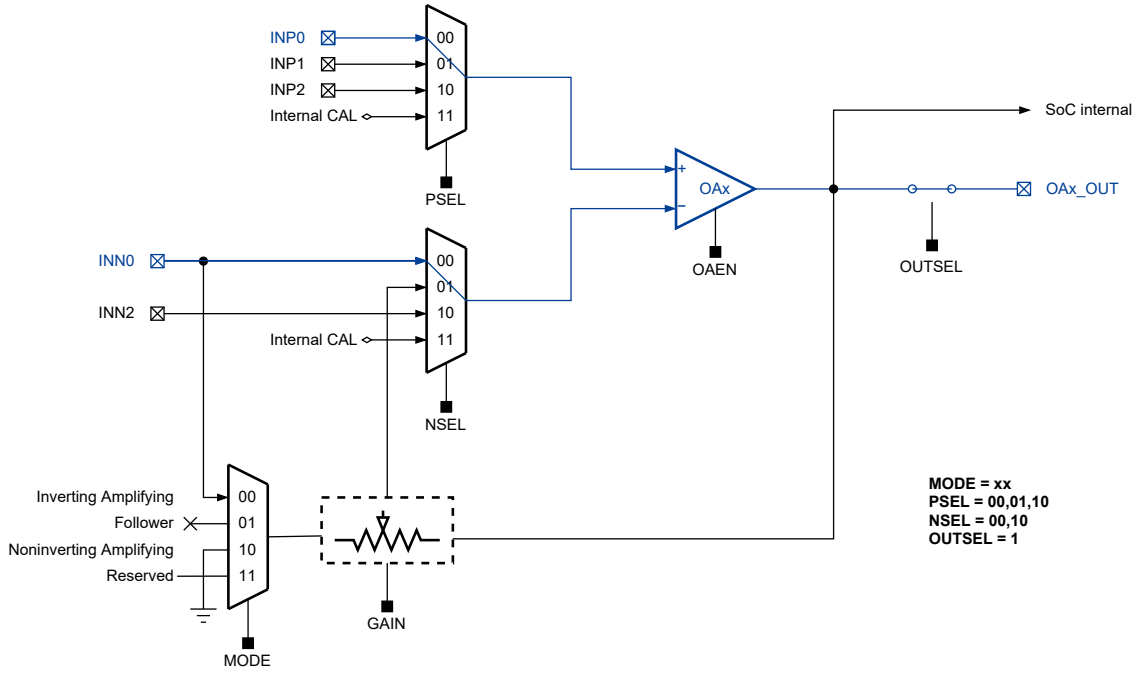


Figure 23-2 OA GP Mode Block Diagram

23.3.2.2 OA Buffer Mode

In this mode, the non-inverting input comes from the external OAxP. The output of the OA is isolated from the R-ladder.

Figure 23-3 shows the example of the OA buffer mode with an OAxP pin as non-inverting input.

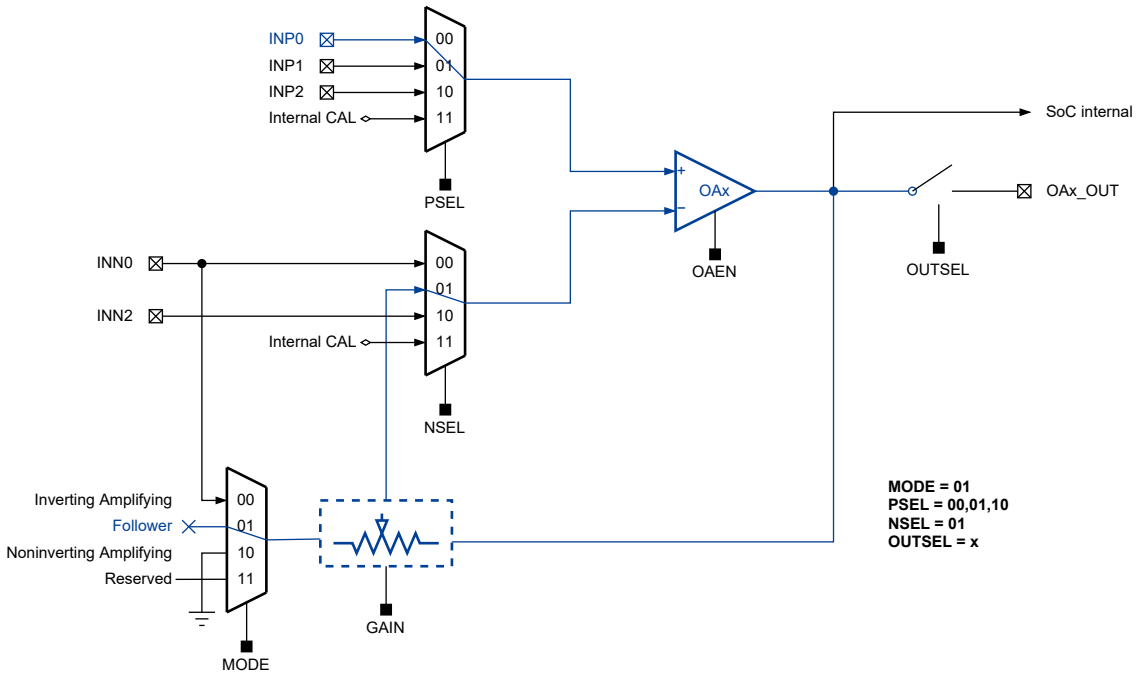


Figure 23-3 Block Diagram of OA Buffer Mode

23.3.2.3 OA PGA Mode

The PGA, also known as Programmable Gain Amplifier, is a circuit component that utilizes a feedback resistance ladder to connect an operational amplifier (OA) output and input as a feedback path. This increases the gain up to 65 times when in PGA mode. Refer to the table below for the various gain settings available in PGA mode.

GAIN	Non-inverting	Inverting
000	2	1
001	3	2
010	5	4
011	9	8
100	17	16
101	26	25
110	33	32
111	65	64

23.3.2.3.1 OA PGA Inverting Mode

In this mode, the inverting input comes from the external OAxINN pin crossing the R-ladder of the PGA. The PGA mode selects OAxINN as the source (MODE = 00), and the OA inverting port must select PGA input (NSEL = 01).

Figure 23-4 shows the example of the OA inverting PGA mode.

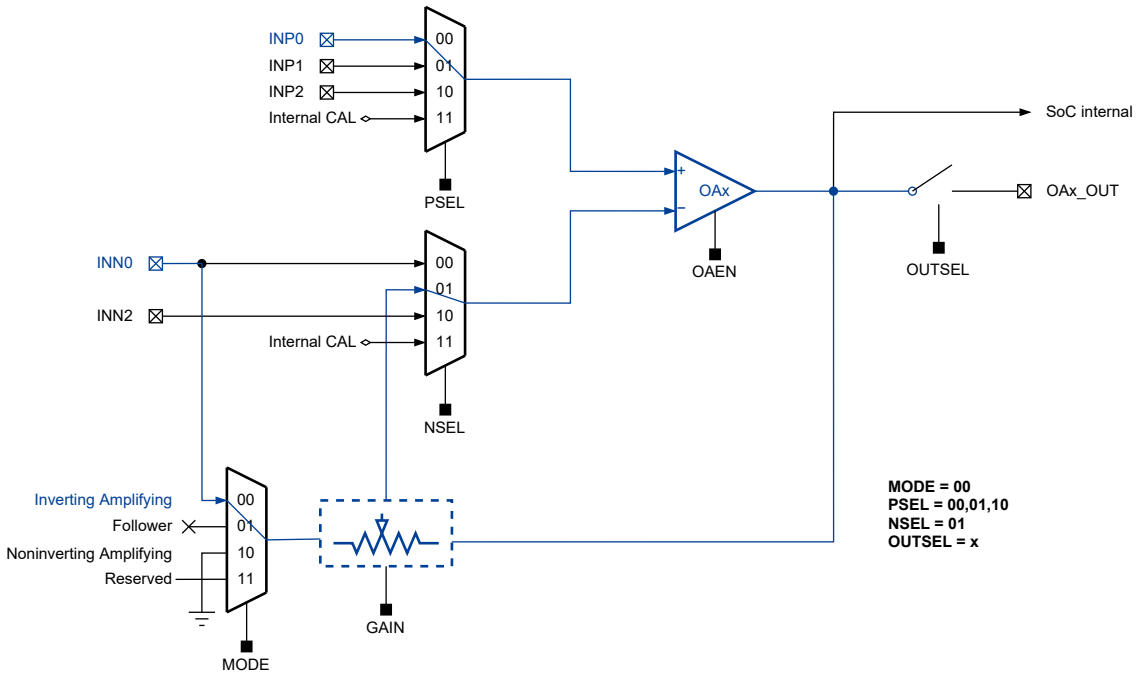


Figure 23-4 Block Diagram of OA PGA Inverting Mode

23.3.2.3.2 OA PGA Non-inverting Mode

In this mode, the non-inverting input is selected from the external pin of OAxP. The OA inverting input must select PGA input (NSEL = 01), and the PGA mode is set to 10 (MODE = 10).

Figure 23-5 shows the example of the OA Non-inverting PGA mode.

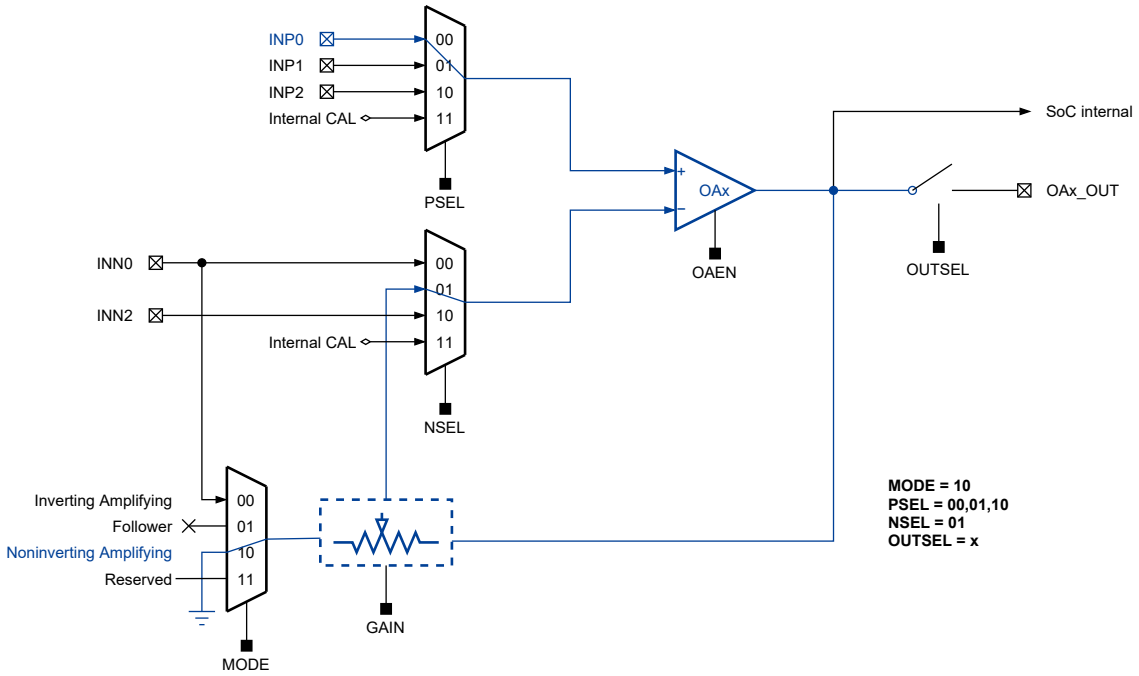


Figure 23-5 Block Diagram of OA PGA Non-inverting Mode

23.3.3 OA Calibration

The OA allows users to calibrate according to the application's requirements. Here are the steps to calibrate the OA:

1. Enable calibration and user trim by setting CALON = 1 and USERTRIM = 1.
2. Set the calibration input voltage by configuring the CALSEL bits. For example, set CALSEL = 10 (VCM = 0.5x VDDA).
3. Write default value 0 to both OSPSET and OSNSET bits, that is, OSPSET = 00000 and OSNSET = 00000.
4. Wait for 1ms, then read the CALOUT value.
   - If CALOUT = 0, write OSPSET bit 4 = 0 and OSNSET bit 4 = 1.
   - If CALOUT = 1, write OSPSET bit 4 =1 and OSNSET bit 4 = 0.

   Repeat this step five times until bit 4 to bit 0 are all calibrated.
5. The calibration is done, and new trim values are written to OSPSET and OSNSET. The calibration value is valid immediately.
6. Set CALON = 0 to return to normal mode.

NOTE: The trim value will be overwritten to the factory value after a reset.

23.3.4 OA Auto Protection

The LOCK bit is a protection bit in OA registers to prevent any unexpected changes. When LOCK = 1, all write/read registers are switched to read-only. The LOCK bit can only be cleared by SYSRESET.

For easier use of OA, several flags indicate forbidden configuration to end users. Refer to Registers for more details.

23.4 Registers

23.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	OA_CSR	Global Control Register

23.4.2 Register Field Details

23.4.2.1 OA_CSR

0x0000			Global Control Register											OA_CSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	LOCK	CALOUT	MODESELERR	GPMODEERR	INMSELERR	INPSELERR	OSPSET					OSNSET				
Type	RW	RO	RO	RO	RO	RO	RW					RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OUTSEL	GAIN		INNSEL		INPSEL		MODE		CALSEL		Reserved	USERTRIM	CALON	OAEN	
Type	RW	RW		RW		RW		RW		RW		RO	RW	RW	RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-2 Global Control Register Description

Field	Name	Description
31	LOCK	0: CSR is read-write 1: CSR is read-only

Field	Name	Description
30	CALOUT	OA calibration status output
29	MODESELERR	Mode selection error flag
28	GPMODEERR	GP mode error flag
27	INMSELERR	INMSEL error flag
26	INPSELERR	INPSEL error flag
25:21	OSPSET	Positive input VCM offset adjustment
20:16	OSNSET	Negative input VCM offset adjustment
15	OUTSEL	0: OA output disable 1: OA output enable
14:12	GAIN	000: Non-inverting gain=2/Inverting gain=-1 001: Non-inverting gain=3/Inverting gain=-2 010: Non-inverting gain=5/Inverting gain=-4 011: Non-inverting gain=9/Inverting gain=-8 100: Non-inverting gain=17/Inverting gain=-16 101: Non-inverting gain=26/Inverting gain=-25 110: Non-inverting gain=33/Inverting gain=-32 111: Non-inverting gain=65/Inverting gain=-64
11:10	INNSEL	00: INN0 01: INN1 10: INN2 11: INN3, used for calibration
9:8	INPSEL	00: INP0 01: INP1 10: INP2 11: INP3, used for calibration

Field	Name	Description
7:6	MODE	00: Inverting Amplifier 01: Buffer 10: Non-Inverting Amplifier 11: Reserved
5:4	CALSEL	00: 0.033*VDDA reference voltage (trim 0.33*VDDA VCM Vos) 01: 0.1*VDDA reference voltage (trim 0.1*VDDA VCM Vos) 10: 0.5*VDDA reference voltage (trim 0.5*VDDA VCM Vos) 11: 0.9*VDDA reference voltage (trim 0.9*VDDA VCM Vos)
3	Reserved	Reserved
2	USERTRIM	User trimming enable This bit allows to switch from 'factory' AOP offset trimmed values to 'user' AOP offset trimmed values. 0: 'factory' trim code used 1: 'user' trim code used
1	CALON	0: Normal mode 1: Calibration mode
0	OAEN	0: OA disable 1: OA enable

Inter-Integrated Circuit (I2C)

This chapter describes the details of the Inter-Integrated Circuit (I2C).

Topics: **Page**

24.1 Introduction

The Inter-Integrated Circuit (I2C) bus interface manages communication between the microcontroller and the serial I2C bus. It enables multi-master capability and controls all I2C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm), and Fast-mode Plus (Fm+).

Additionally, it is compatible with the System Management Bus (SMBus) and Power Management Bus (PMBus). The use of DMA can help reduce CPU overload.

24.2 Features

- I2C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- The following additional features are also available depending on the product implementation
- SMBus Specification rev 3.0 compatibility includes:
  - Hardware Packet Error Checking (PEC) generation and verification with ACK control
  - Command and data acknowledge control
  - Support for Address Resolution Protocol (ARP)
  - Support for host and device
  - SMBus alert functionality
  - Detection of timeouts and idle condition
- PMBus rev 1.3 standard compatibility

24.3 Functional Description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I2C bus.This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported, the additional optional SMBus Alert pin (SMBA) is also available.

24.3.1 Block Diagram

Figure 24-1 shows the block diagram of the I2C interface.

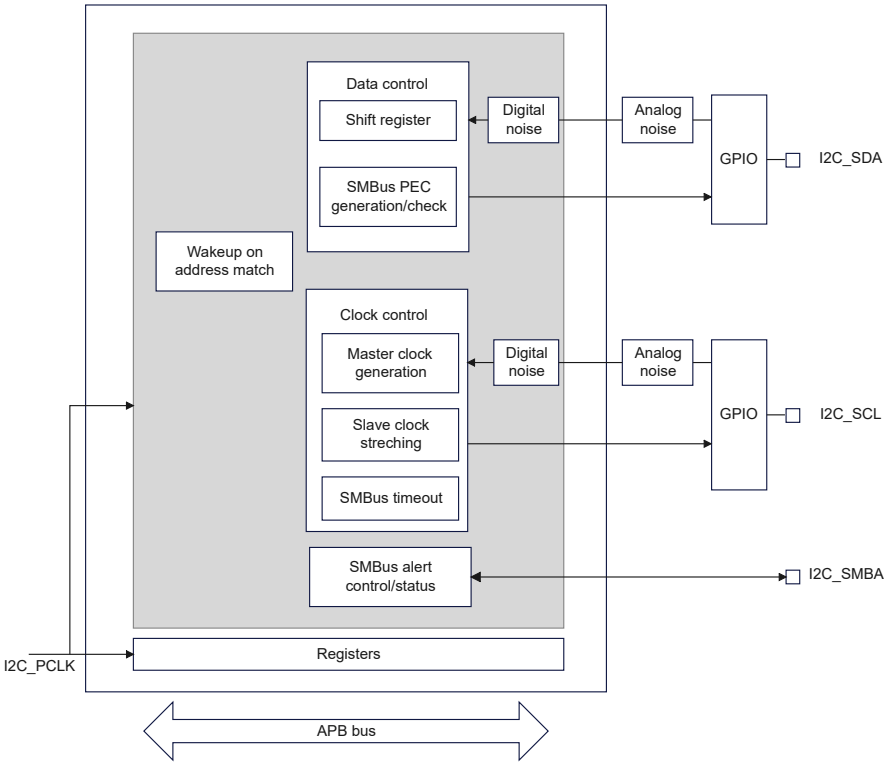


Figure 24-1 I2C Block Diagram

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus (Fm+) operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG).

24.3.2 Pins and Internal Signals

Table 24-1 I2C Input/Output Pins

Pin Name	Signal Type	Description
I2C_SDA	Bidirectional	I2C data
I2C_SCL	Bidirectional	I2C clock
I2C_SMBA	Bidirectional	SMBus alert

Table 24-2 I2C Internal Input/Output Signals

Internal Signal Name	Signal Type	Description
i2c_ker_ck (from i2c_pclk)	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_int	Output	I2C interrupts, refer to Table 24-16 for the full list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

24.3.3 Clock Requirements

The I2C kernel is clocked by I2CCLK. The period of I2CCLK, denoted as t_{I2CCLK}, must satisfy the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$ and $t_{I2CCLK} < t_{HIGH}$

Where:

- t_{LOW}: The low duration of SCL.
- t_{HIGH}: The high duration of SCL.
- $t_{filters}$: When enabled, sum of the delays brought by the analog filter and by the digital filter. The analog filter delay has a maximum value of 260 ns, while the digital filter delay is calculated as DNF x t_{I2CCLK}.

Additionally, the PCLK clock period, denoted as t_{PCLK}, must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} represents the period of SCL.

NOTE: When the I2C kernel is clocked by PCLK, this clock must respect the conditions for t_{I2CCLK}.

24.3.4 Mode Selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it 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, allowing multi-master capability.

Communication Flow

In master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first bytes 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 Figure 24-2 for more details.

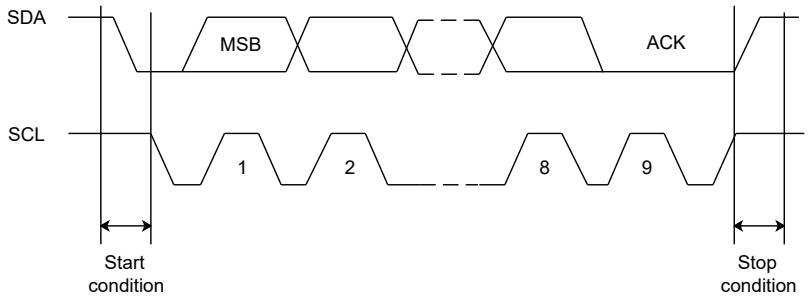


Figure 24-2 I2C Bus Protocol

Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

24.3.5 I2C Initialization

24.3.5.1 Enabling and Disabling Peripheral

The I2C peripheral clock must be configured and enabled in the clock controller. Then the I2C can be enabled by setting the I2CEN bit in the I2C_CR1 register.

When the I2C is disabled (I2CEN = 0), the I2C performs a software reset. Refer to Software Reset for more details.

24.3.5.2 Noise Filters

Before enabling the I2C peripheral by setting the I2CEN bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I2C specification which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows spikes with a programmable length of 1 to 15 I2CCLK periods to be suppressed.

Table 24-3 Analog vs. Digital Filters

	Pulse Width of Suppressed Spikes	Benefits	Drawbacks
Analog filter	≤ 50 ns	Available in stop mode	Variation vs. temperature, voltage, process
Digital filter	Programmable length from 1 to 15 I2C peripheral clocks	• Programmable length: Extra filtering capability versus standard requirements • Stable length	

NOTE: Changing the filter configuration is not allowed when the I2C is enabled.

24.3.5.3 I2C Timings

The timings must be configured to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PREDIV[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

If the data is already available in the I2C_TXDR register, the data will be sent on SDA after the SDADEL delay, as illustrated in Figure 24-3.

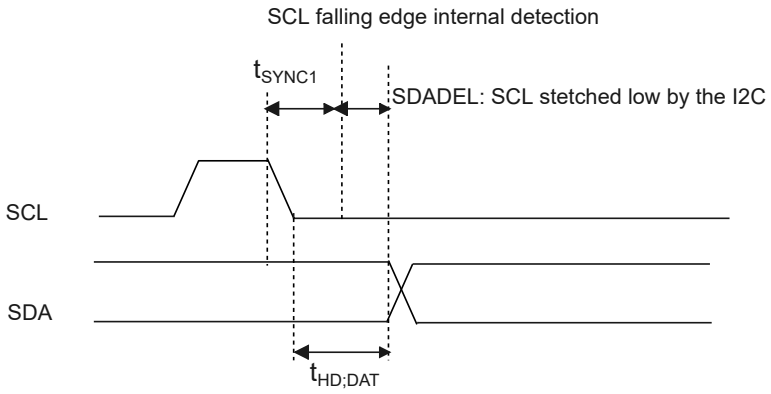


Figure 24-3 Data Hold Time

The SCLDEL counter starts when the data is sent on SDA output, as illustrated in Figure 24-4.

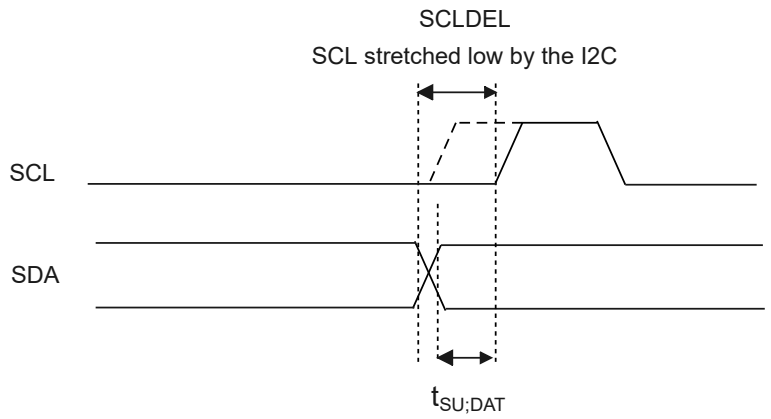


Figure 24-4 Data Setup Time

When the internal detection of the falling edge of the SCL occurs, a delay is introduced before the SDA output is sent.

This delay is calculated as t_{SDADEL}= SDADEL x t_{PREDIV} + t_{I2CCLK}, where t_{PREDIV} = (PREDIV+1) x t_{I2CCLK}. t_{SDADEL} impacts the hold time, $t_{HD;DAT}$.

The total SDA output delay can be calculated as follows:

t_{SYNC1} + {[SDADEL x (PREDIV+1) + 1] x t_{I2CCLK}}

t_{SYNC1} duration depends on these parameters:

- SCL falling slope
- Input delay brought by the analog filter when enabled: $t_{AF(min)} < t_{AF} < t_{AF(max)}$

- Input delay brought by the digital filter when enabled: t_{DNF} = DNF x t_{I2CCLK}
- Delay caused by synchronizing SCL to the I2CCLK clock (2 to 3 I2CCLK periods)

To bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

$\{t_{f(max)} + t_{HD;DAT\ (min)} - t_{AF(min)} - [(DNF +3) \times t_{I2CCLK}]\} / \{(PREDIV +1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF+4) \times t_{I2CCLK}]\} / \{(PREDIV +1) \times t_{I2CCLK}\}$

NOTE: $t_{AF(min)}$ / $t_{AF(max)}$ is part of the equation only when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The maximum $t_{HD;DAT}$ can be 3.45 µs, 0.9 µs and 0.45 µs for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - 260\ ns - [(DNF+4) \times t_{I2CCLK}]\} / \{(PREDIV +1) \times t_{I2CCLK}\}$.

NOTE: This condition can be violated when NOEXT = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.

Refer to Table 24-4 for t_f, t_r, $t_{HD;DAT}$ and $t_{VD;DAT}$ standard values.

After t_{SDADEL} delay, or after sending SDA output in case the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, SCL line is kept at low level during the setup time. This setup time is t_{SCLDEL} = (SCLDEL+1) x t_{PREDIV}, where t_{PREDIV} = (PREDIV+1) x t_{I2CCLK}.

t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PREDIV+1)] \times t_{I2CCLK}]\} - 1 \leq SCLDEL$

Refer to Table 24-4 for t_r and $t_{SU;DAT}$ standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

NOTE: At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, in case the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOEXT = 1 in slave mode, the SCL is not stretched. Consequently the SDADEL must be programmed in such a way to guarantee also a sufficient setup time.

Table 24-4 I2C-SMBus Specification Data Setup and Hold Times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	μs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PREDIV[3:0], SCLHI[7:0] and SCLLOW[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is t_{SCLLOW} = (SCLLOW+1) x t_{PREDIV} where t_{PREDIV} = (PREDIV + 1) x t_{I2CCLK}.

  t_{SCLLOW} impacts the SCL low time t_{LOW}.
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is t_{SCLHI} = (SCLHI+1) x t_{PREDIV} where t_{PREDIV} = (PREDIV+1) x t_{I2CCLK}. t_{SCLHI} impacts the SCL high time t_{HIGH}.

Refer to Initialization for more details.

NOTE: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOEXT mode must also be configured before enabling the peripheral. Refer to Initialization for more details.

NOTE: Changing the NOEXT configuration is not allowed when the I2C is enabled.

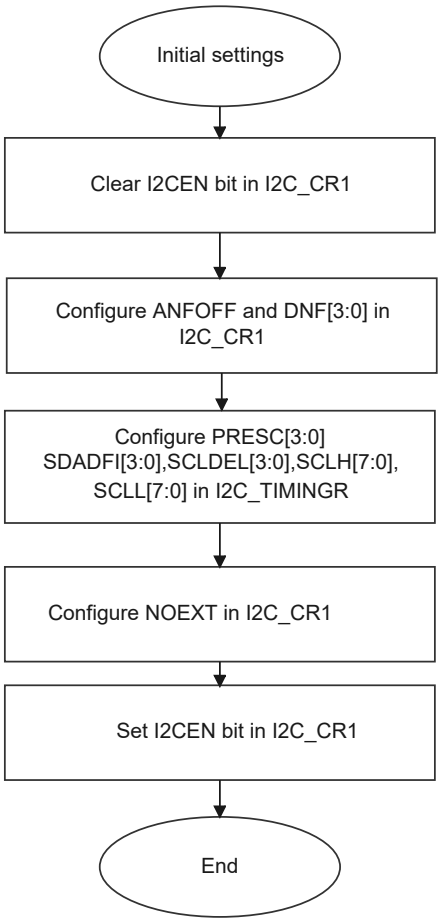


Figure 24-5 I2C Initialization Flowchart

24.3.6 Software Reset

A software reset can be initiated by clearing the I2CEN bit in the I2C_CR1 register. In that case, I2C lines SCL and SDA are released. Internal state machines are reset, and both the communication control bits and status bits come back to their reset value. However, this process does not affect the configuration registers.

The following register bits are affected:

- I2C_CR2 register: START, STOP, NACK
- I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVF

Furthermore, when the SMBus feature is supported, the following additional register bits are affected:

- I2C_CR2 register: PECBYTE
- I2C_ISR register: PECERR, TIMEOUT, ALERT

I2CEN must be kept low during at least 3 APB clock cycles to perform the software reset. This is ensured by writing the following software sequence:

- Write I2CEN = 0.
- Check I2CEN = 0.
- Write I2CEN = 1.

24.3.7 Data Transfer

The data transfer is managed through transmit and receive data registers and a shift register.

24.3.7.1 Data Reception

The shift register gets filled by the SDA input. Once the 8th SCL pulse is received, indicating the complete data byte has been received, the shift register's contents are copied into the I2C_RXDR register, provided it is empty (RXNE = 0). If RXNE = 1, it implies that the previously received data byte has not been read yet. In response to this, the SCL line is held low until the I2C_RXDR is read. This hold, or stretch, is implemented between the 8th and 9th SCL pulse, which is before the acknowledge pulse.

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the acknowledge pulse).

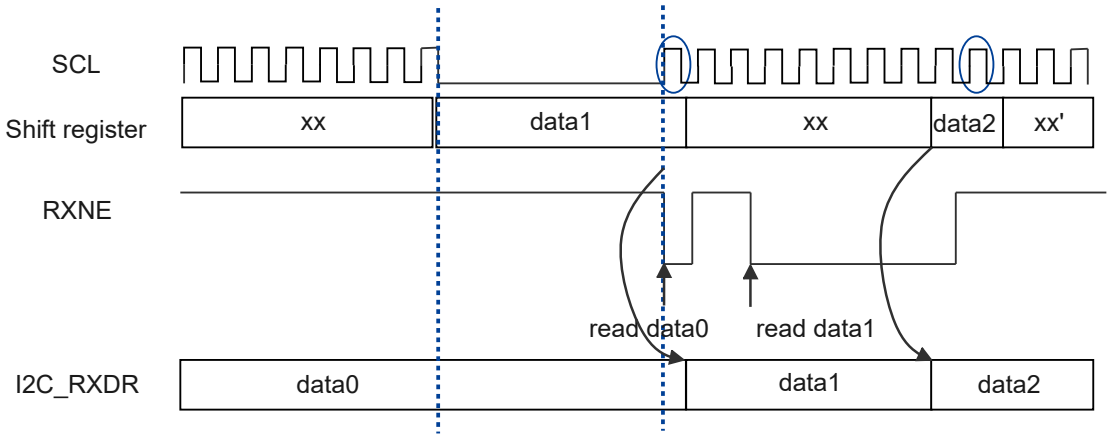


Figure 24-6 Data Reception

24.3.7.2 Data Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE = 1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the 9th SCL pulse.

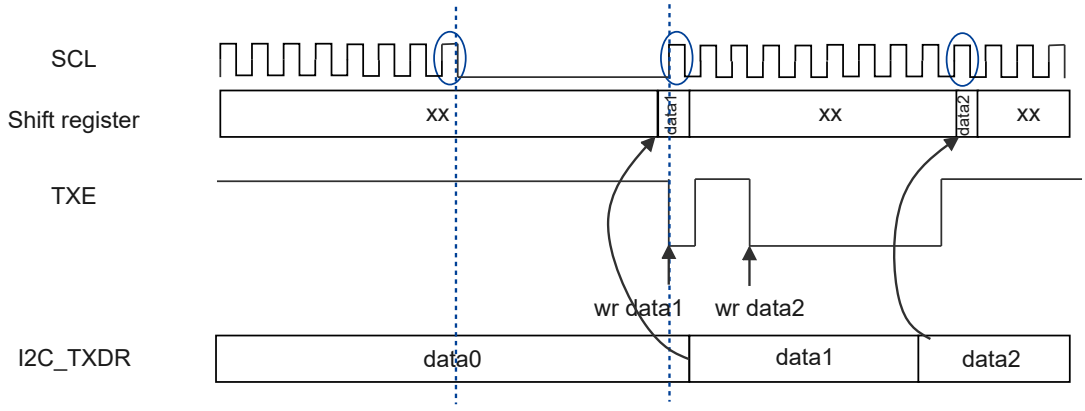


Figure 24-7 Data Transmission

24.3.7.3 Hardware Transfer Management

The I2C includes a hardware-based byte counter for managing byte transfer and controlling communication in different modes. These modes include generating NACK, STOP, and RESTART in master mode, as well as controlling ACK in slave receiver mode and performing PEC generation/checking when the SMBus feature is supported.

The byte counter is always used in master mode. By default, it is disabled in slave mode, but it can be enabled through software by setting the SBCEN (Slave Byte Control) bit in the I2C_CR1 register.

The number of bytes to be transferred (NBYTE) is programmed in the NBYTE[7:0] bit field in the I2C_CR2 register. If NBYTE is greater than 255, or if a receiver needs to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTE is transferred, and an interrupt is generated if TCIE is set. The SCL is stretched as long as TCR flag is set. The TCR is cleared by software when NBYTE is written to a non-zero value.

In master mode, when RELOAD = 0, the counter can be used in 2 modes:

- Automatic end mode (AUTOEND = 1 in the I2C_CR2 register)

  In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTE[7:0] bit field is transferred.
- Software end mode (AUTOEND = 0 in the I2C_CR2 register)

  In this mode, software action is expected once the number of bytes programmed in the NBYTE[7:0] bit field is transferred. The TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when either the START or STOP bit is set in the I2C_CR2 register.

  This mode must be used when the master wants to send a RESTART condition.

NOTE: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 24-5 I2C Configuration

Function	SBC Bit	RELOAD Bit	AUTOEND Bit
Master Tx/Rx NBYTE + STOP	x	0	1
Master Tx/Rx + NBYTE + RESTART	x	0	0
Slave Tx/Rx all received bytes ACKed	0	x	x
Slave Rx with ACK control	1	1	x

24.3.8 I2C Slave Mode

24.3.8.1 Initialization

To work in slave mode, the user must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.

  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.

- If additional slave addresses are required, the 2nd slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK = 0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.

- The general call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOEXT = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C_ISR register to check which address matched. DIR flag must also be checked to know the transfer direction.

24.3.8.2 Slave Clock Stretching

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCR bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE = 1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC = 1 and RELOAD = 1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTE[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during [(SDADEL+SCLDEL+1) x (PREDIV+1) + 1] x t_{I2CCLK}.

Slave without Clock Stretching (NOEXT = 1)

When NOEXT = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVF flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVF flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVF status is provided, even for the first data to be transmitted.

- In reception, the data must be read from the I2C_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVF flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

24.3.8.3 Slave Byte Control Mode

To allow byte ACK control in slave reception mode, The Slave byte control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

The Reload mode must be selected to allow byte ACK control in slave reception mode (RELOAD = 1). To get control of each byte, NBYTE must be initialized to 0x01 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTE to a non-zero value: the acknowledge or not- acknowledge is sent and next byte can be received.

NBYTE can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTE data reception.

NOTE: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

NOTE: The slave byte control mode is incompatible with the NOEXT mode. Configuring SBC when NOEXT = 1 is prohibited.

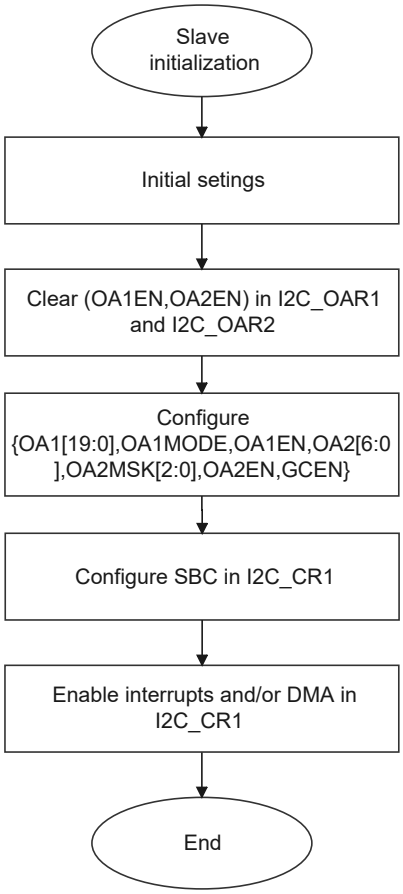


Figure 24-8 Slave Initialization Flowchart

24.3.8.4 Slave Transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The slave automatically releases the SCL and SDA lines to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to '0'. In this case, If TXE = 0 when the slave address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit to program a new data byte.

In Slave byte control mode (SBC = 1), the number of bytes to be transmitted must be programmed in NBYTE in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTE.

NOTE: When NOEXT = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine, to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit to program a new data byte. The STOPF bit must be cleared only after these actions, to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVF flag is set).

  If a TXIS event is needed, (transmit interrupt or transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, to generate a TXIS event.

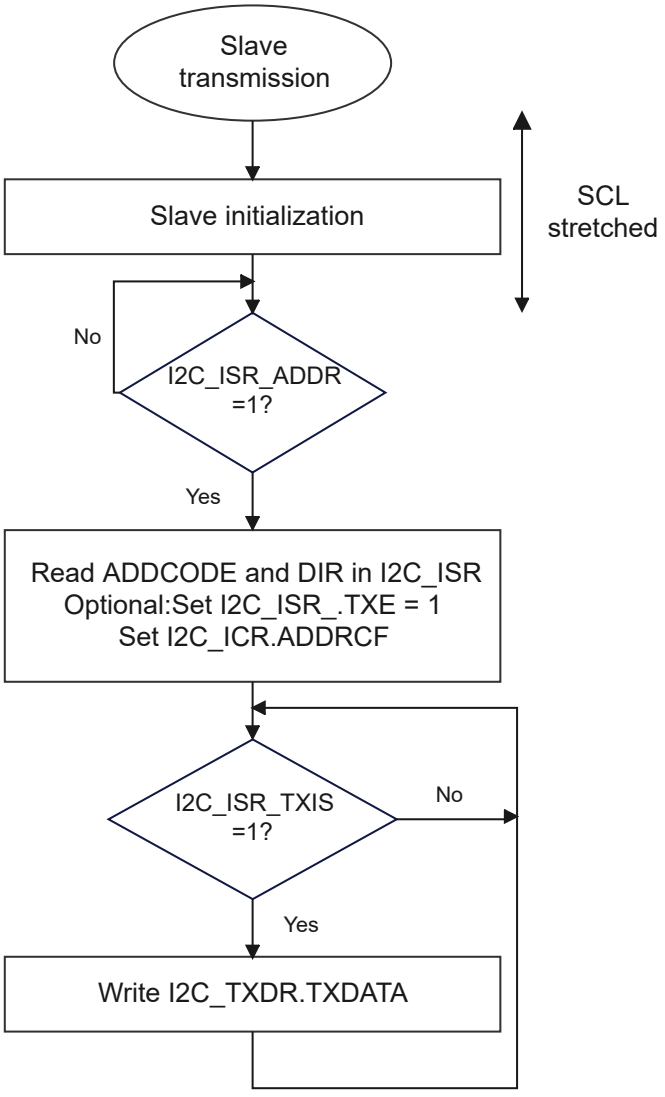


Figure 24-9 Transfer Sequence Flowchart for I2C Slave Transmitter, NOEXT = 0

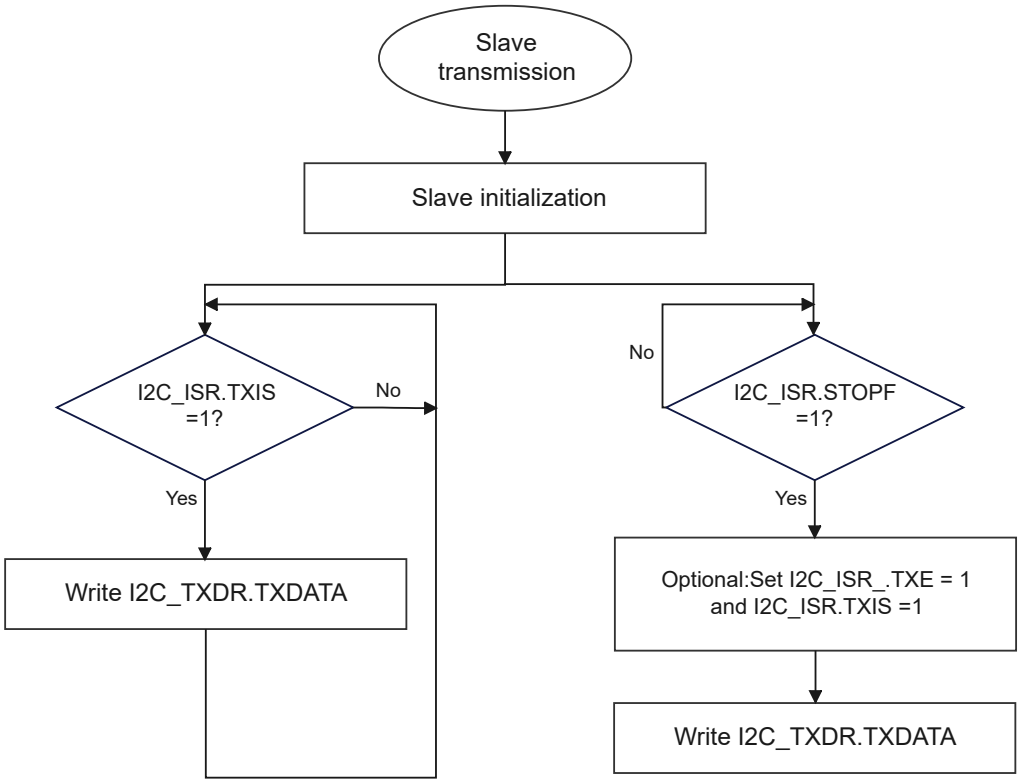


Figure 24-10 Transfer Sequence Flowchart for I2C Slave Transmitter, NOEXT = 1

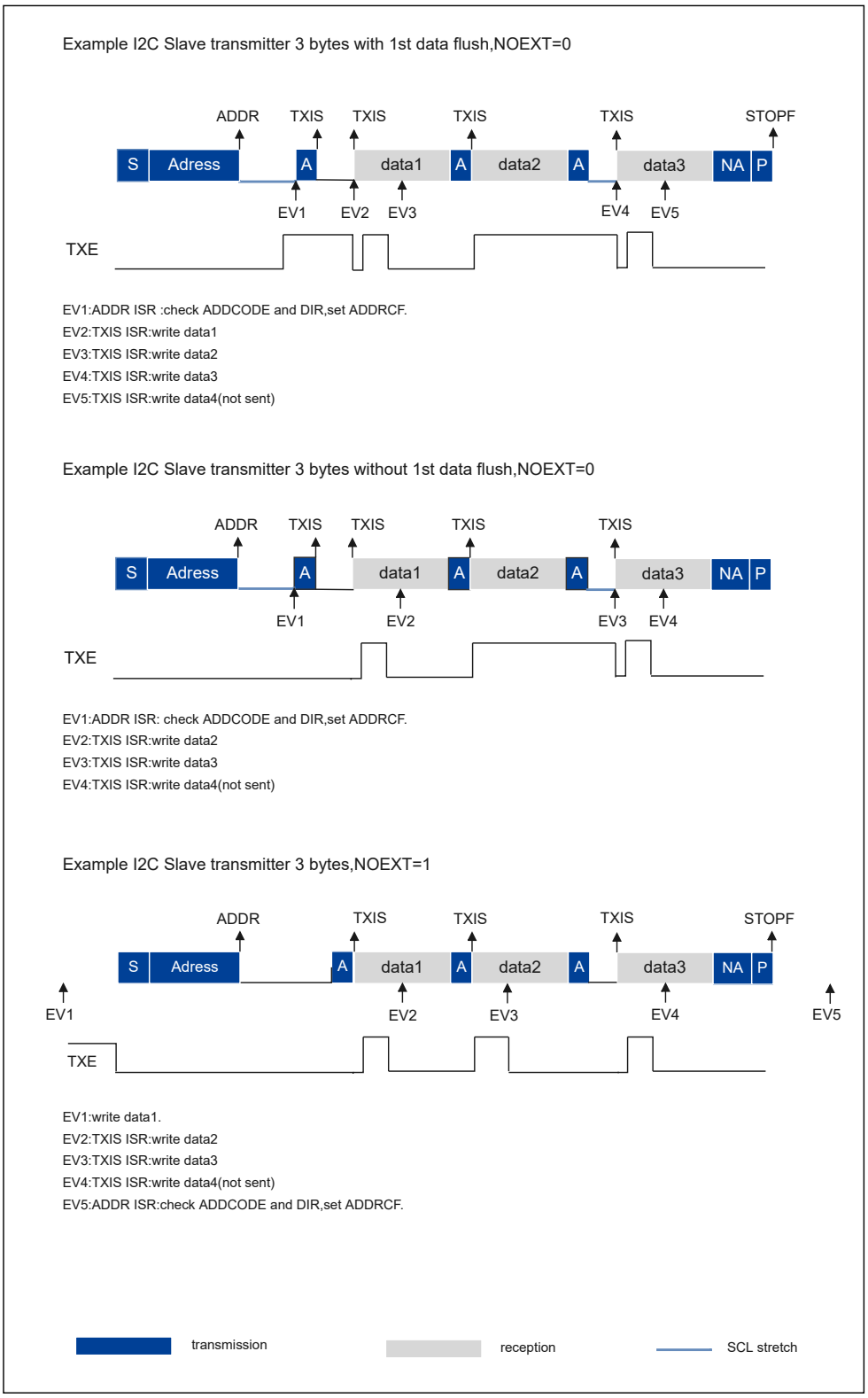


Figure 24-11 Transfer Bus Diagrams for I2C Slave Transmitter

24.3.8.5 Slave Receiver

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

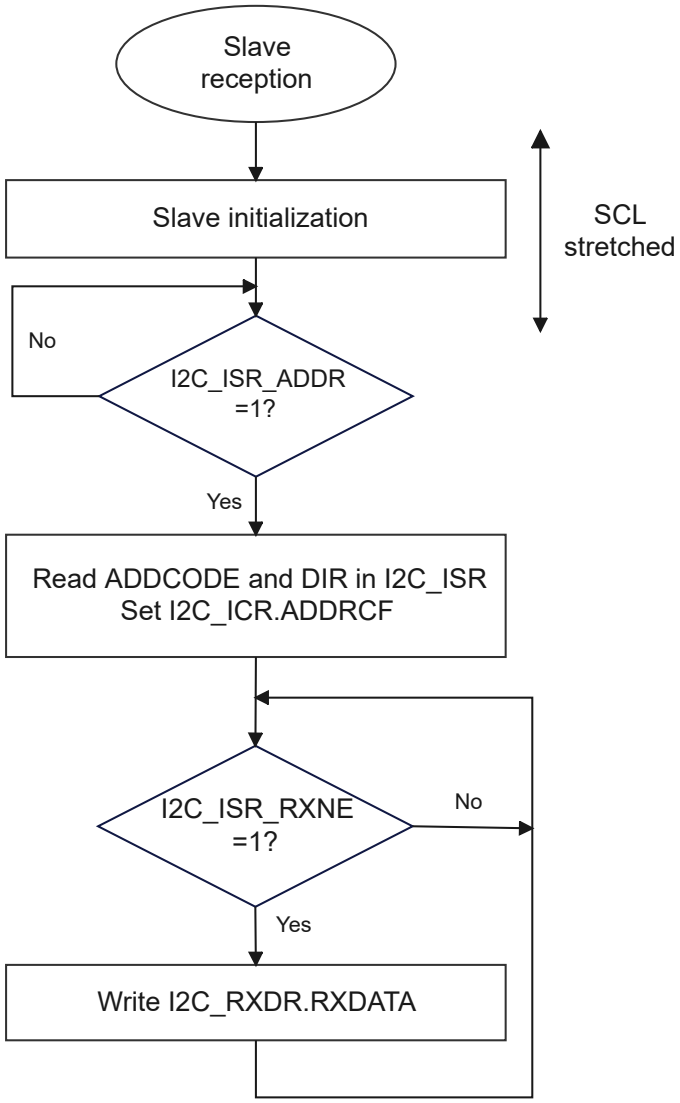


Figure 24-12 Transfer Sequence Flowchart for I2C Slave Receiver, NOEXT = 0

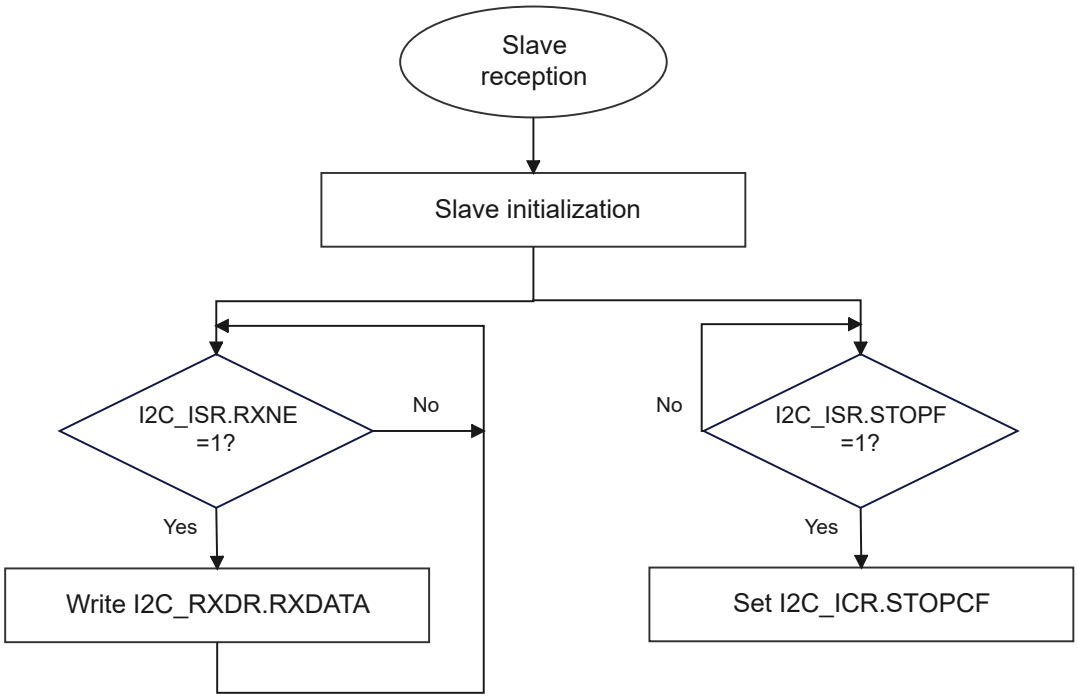


Figure 24-13 Transfer Sequence Flowchart for I2C Slave Receiver, NOEXT = 1

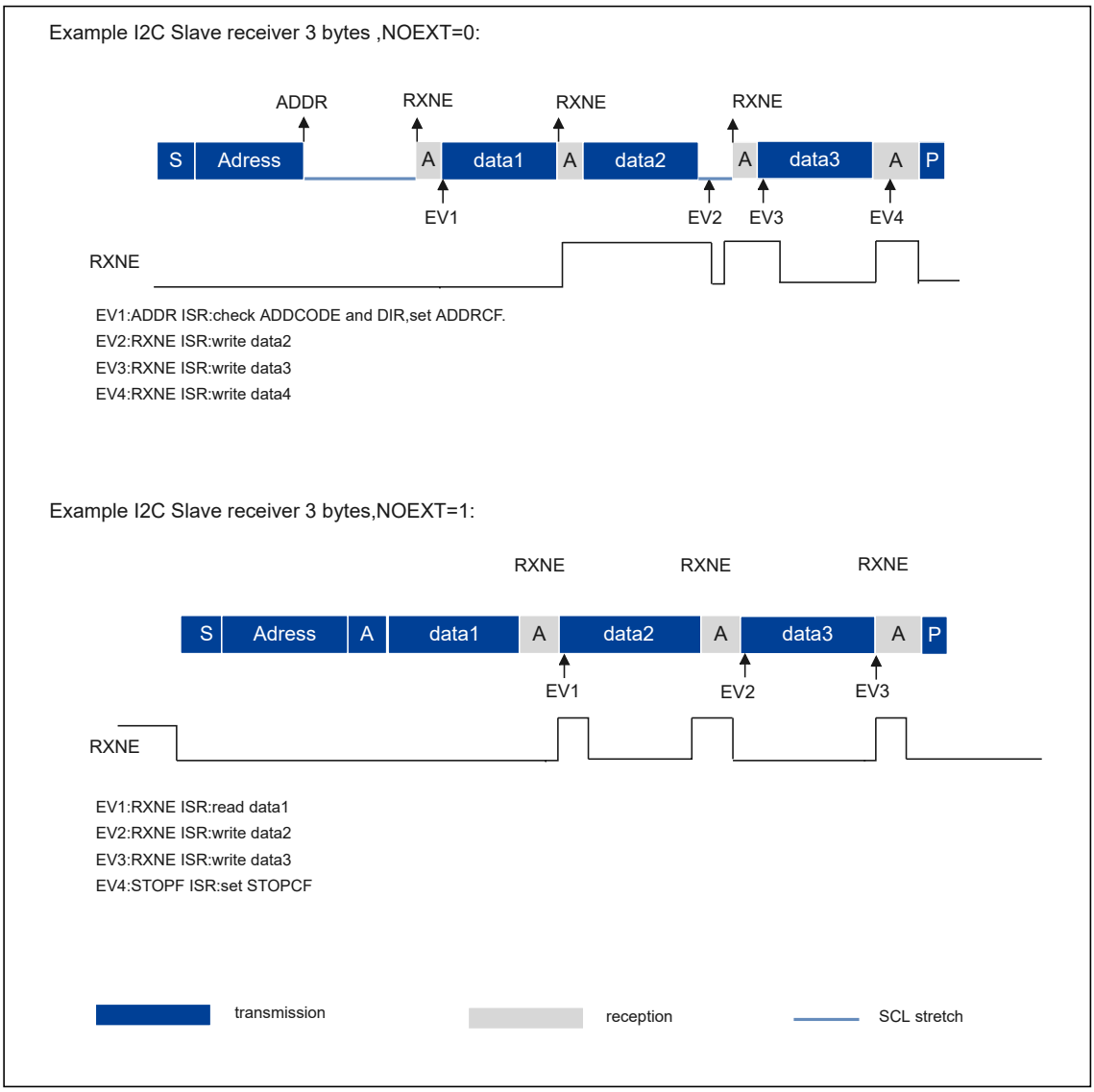


Figure 24-14 Transfer Bus Diagrams for I2C Slave Receiver

24.3.9 I2C Master Mode

24.3.9.1 Initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLHI and SCLLOW bits in the I2C_TIMINGR register.

A clock synchronization mechanism is implemented to support multi-master environment and slave clock stretching.

To allow clock synchronization:

- The low level of the clock is counted using the SCLLOW counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLHI counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases

SCL to high level once the SCLLOW counter reaches the value programmed in the SCLLOW[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLHI counter is reached reaches the value programmed in the SCLHI[7:0] bits in the I2C_TIMINGR register.

The master clock period is calculated as:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLHI+1) + (SCLLOW+1)] \times (PREDIV+1) \times t_{I2CCLK}\}$

The duration of t_{SYNC1} depends on these parameters:

- SCL falling slope
- Input delay from analog filter, if enabled
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of t_{SYNC2} depends on these parameters:

- SCL rising slope
- Input delay from analog filter, if enabled
- When enabled, input delay induced by the digital filter: DNF x t_{I2CCLK}
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

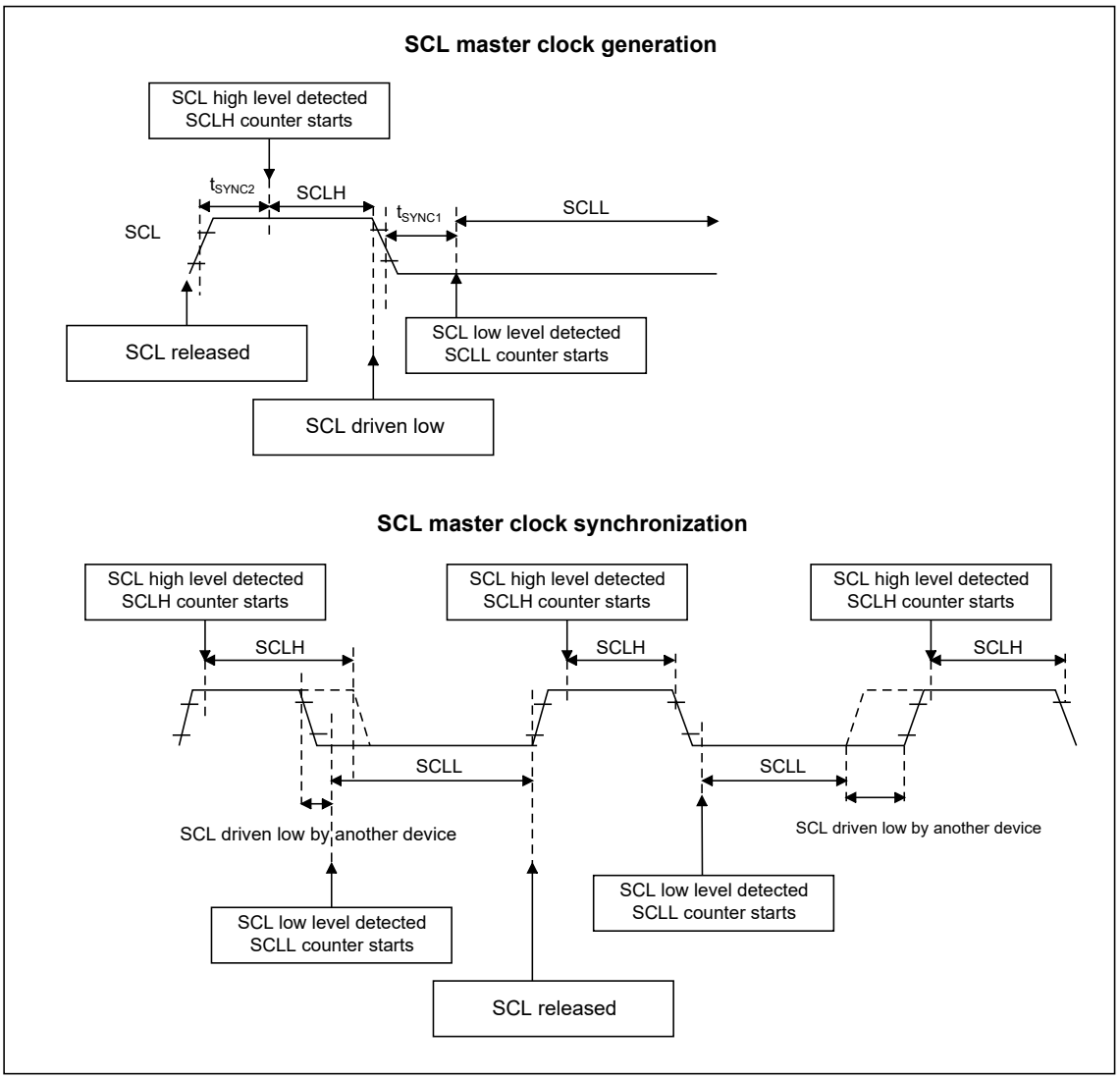


Figure 24-15 Master Clock Generation

NOTE: To comply with I2C or SMBus Specifications, it is important to follow the timings provided in Table 24-6 for the master clock.

Table 24-6 I2C-SMBus Specification Clock Timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	μs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	μs
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	μs

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	μs
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	μs
t_{HIGH}	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	μs
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	ns

NOTE:
- SCLL is also used to generate the t_{BUF} and $t_{SU:STA}$ timings.
- SCLH is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.
- Refer to I2C_TIMINGR Register Configuration Example for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

24.3.9.2 Master Communication Initialization (Address Phase)

To initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- The number of bytes to be transferred: NBYTE[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTE[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of tBUF.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

NOTE: The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.

In 10-bit addressing mode, when the Slave Address first 7 bits is NACKed by the slave, the master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCR must be set if a NACK is received from the slave, to stop sending the slave address.

If the I2C is addressed as a slave (ADDR = 1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared, when the ADDRCR bit is set.

NOTE: The same procedure is applied for a Repeated Start condition. In this case BUSY = 1.

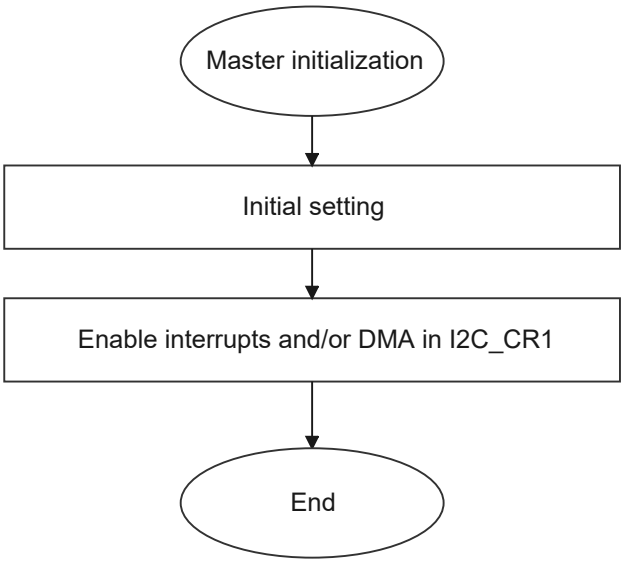


Figure 24-16 Master Initialization Flowchart

24.3.9.3 Initialization of Master Receiver Addressing 10-bit Address Slave

If the master addresses a 10-bit address slave, it first transmits data to the slave and then reads data from it. A master transmission flow must be done first. Then, a repeated start is set with the 10-bit slave address configured. In this case, the master sends this sequence: Restart + Slave address 10-bit header read.

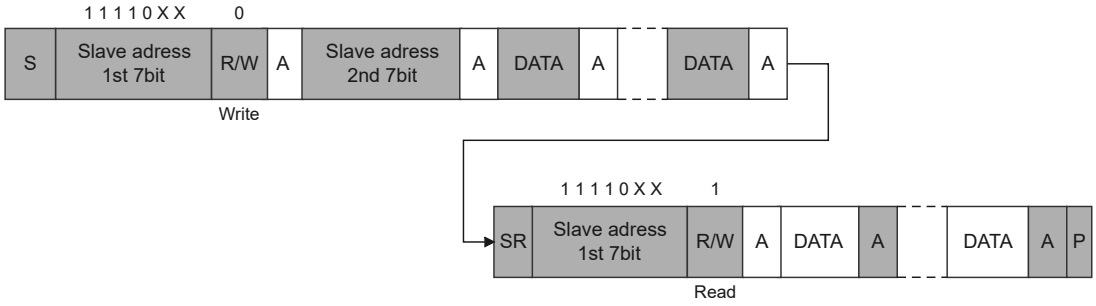


Figure 24-17 10-bit Address Read Access

24.3.9.4 Master Transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTE[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTE data have been

transferred, the TCR flag is set and the SCL line is stretched low until NBYTE[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD = 0 and NBYTE data have been transferred:
  - In automatic end mode (AUTOEND = 1), a STOP is automatically sent.
  - In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low to perform software actions:

A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.

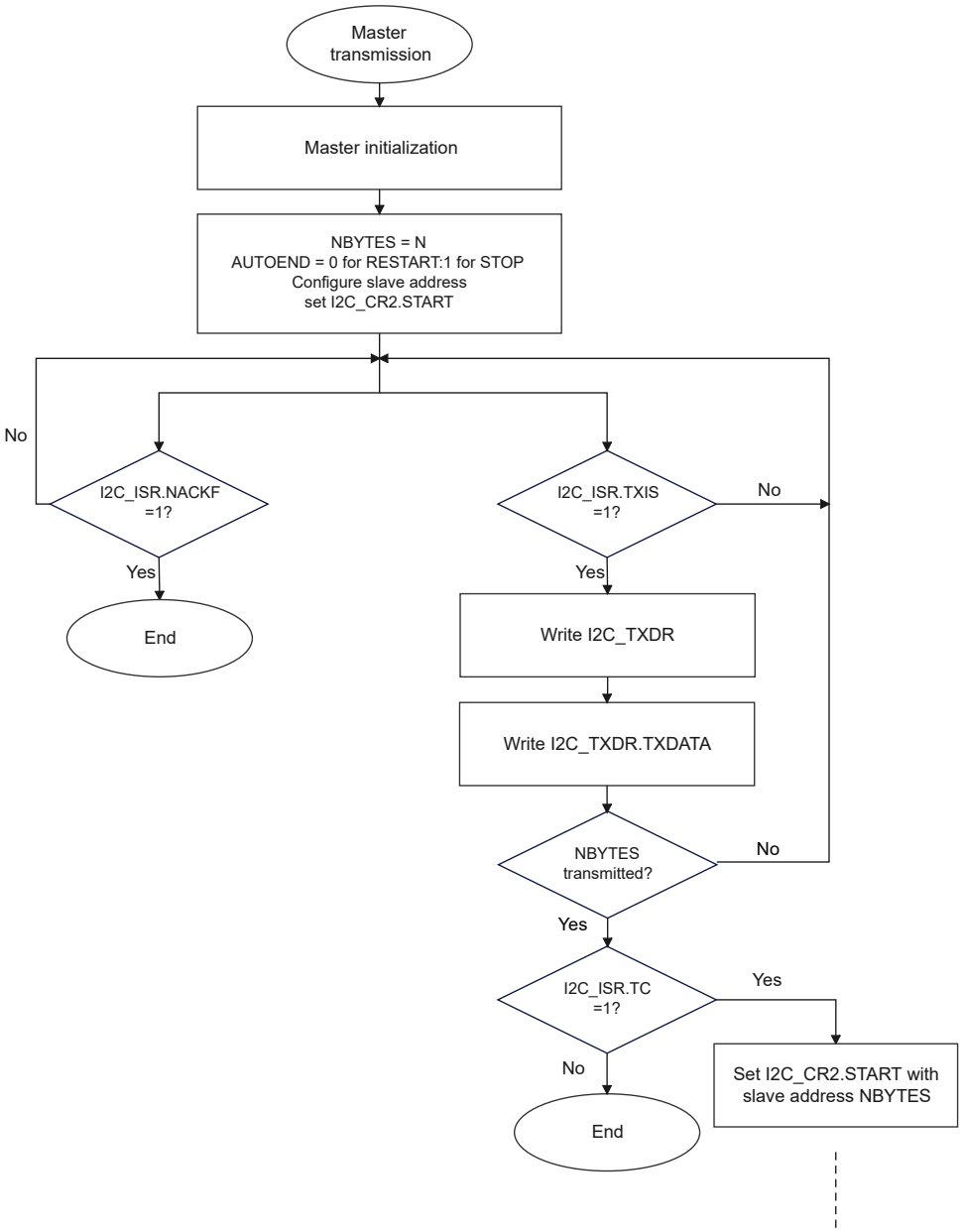


Figure 24-18 Transfer Sequence Flowchart for I2C Master Transmitter for N ≤ 255 Bytes

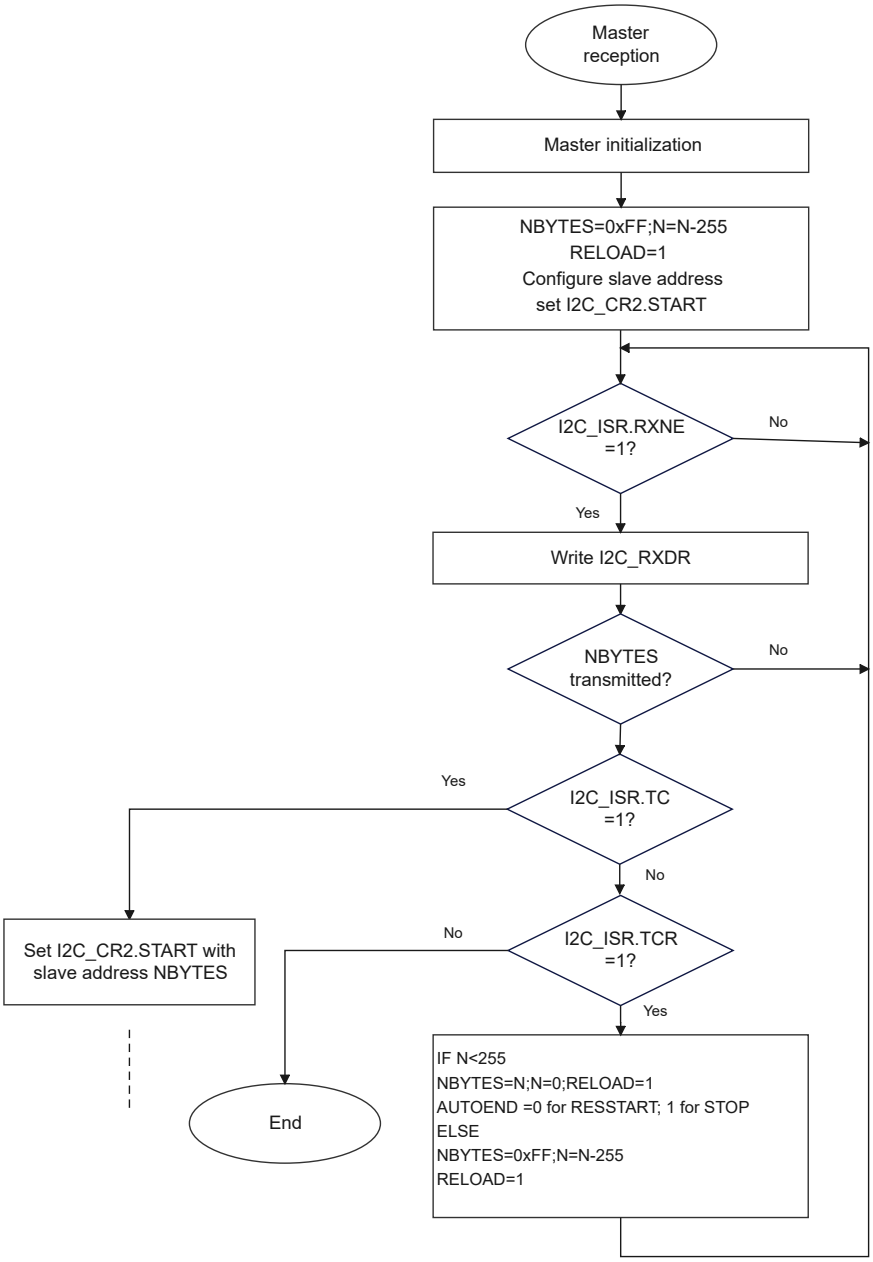


Figure 24-19 Transfer Sequence Flowchart for I2C Master Transmitter for N > 255 Bytes

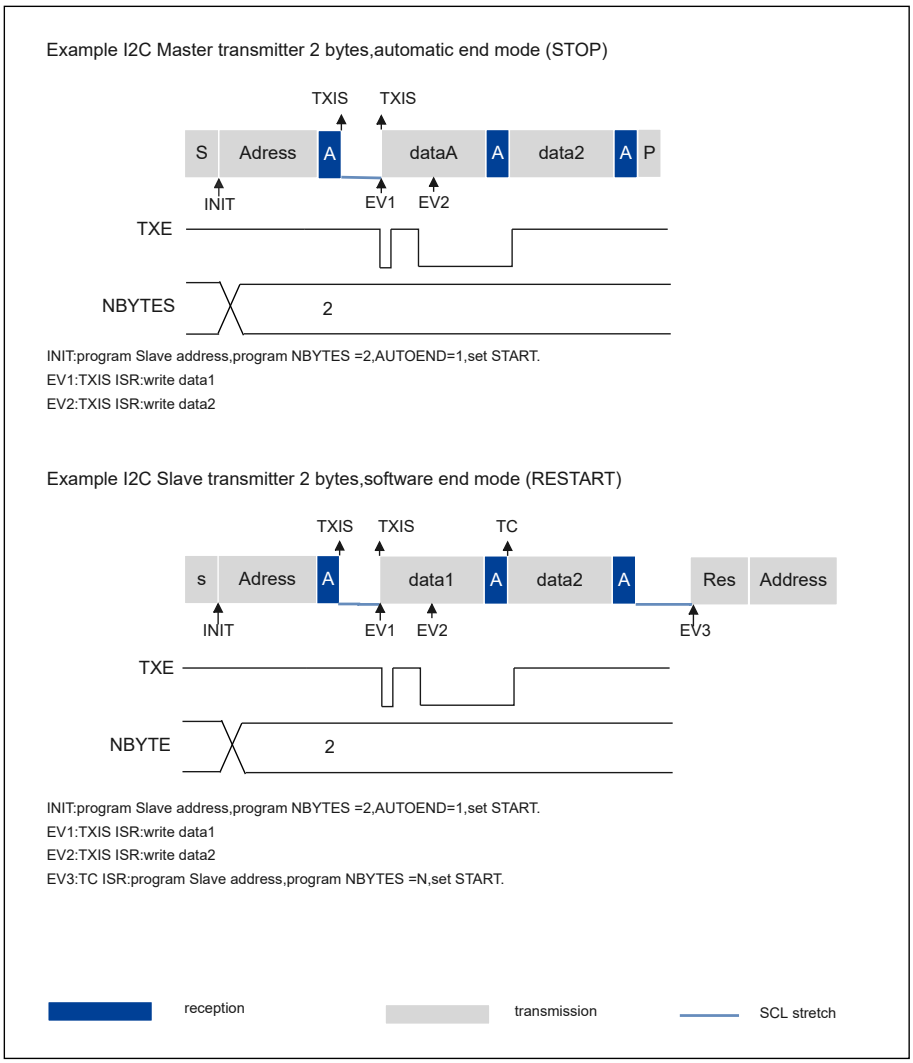


Figure 24-20 Transfer Bus Diagrams for I2C Master Transmitter

24.3.9.5 Master Receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTE[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTE[7:0] is written to a non-zero value.

When RELOAD = 0 and NBYTE[7:0] data have been transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low to allow software actions:

  A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

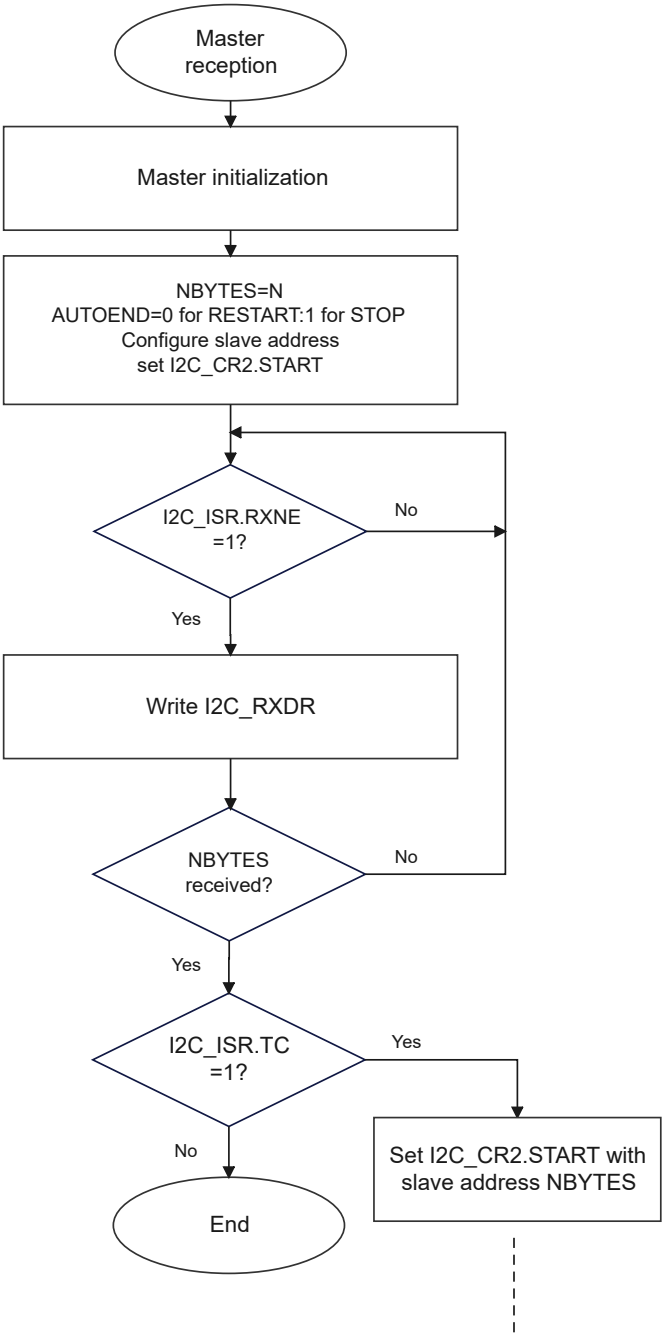


Figure 24-21 Transfer Sequence Flowchart for I2C Master Receiver for N ≤ 255 Bytes

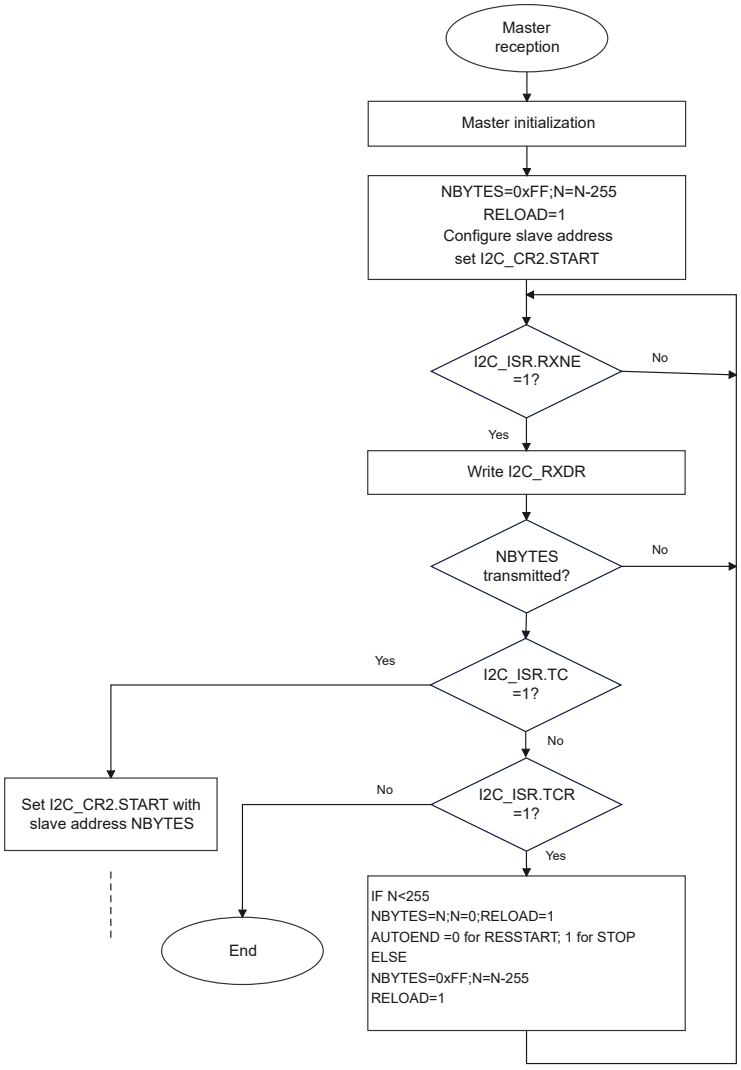


Figure 24-22 Transfer Sequence Flowchart for I2C Master Receiver for N>255 Bytes

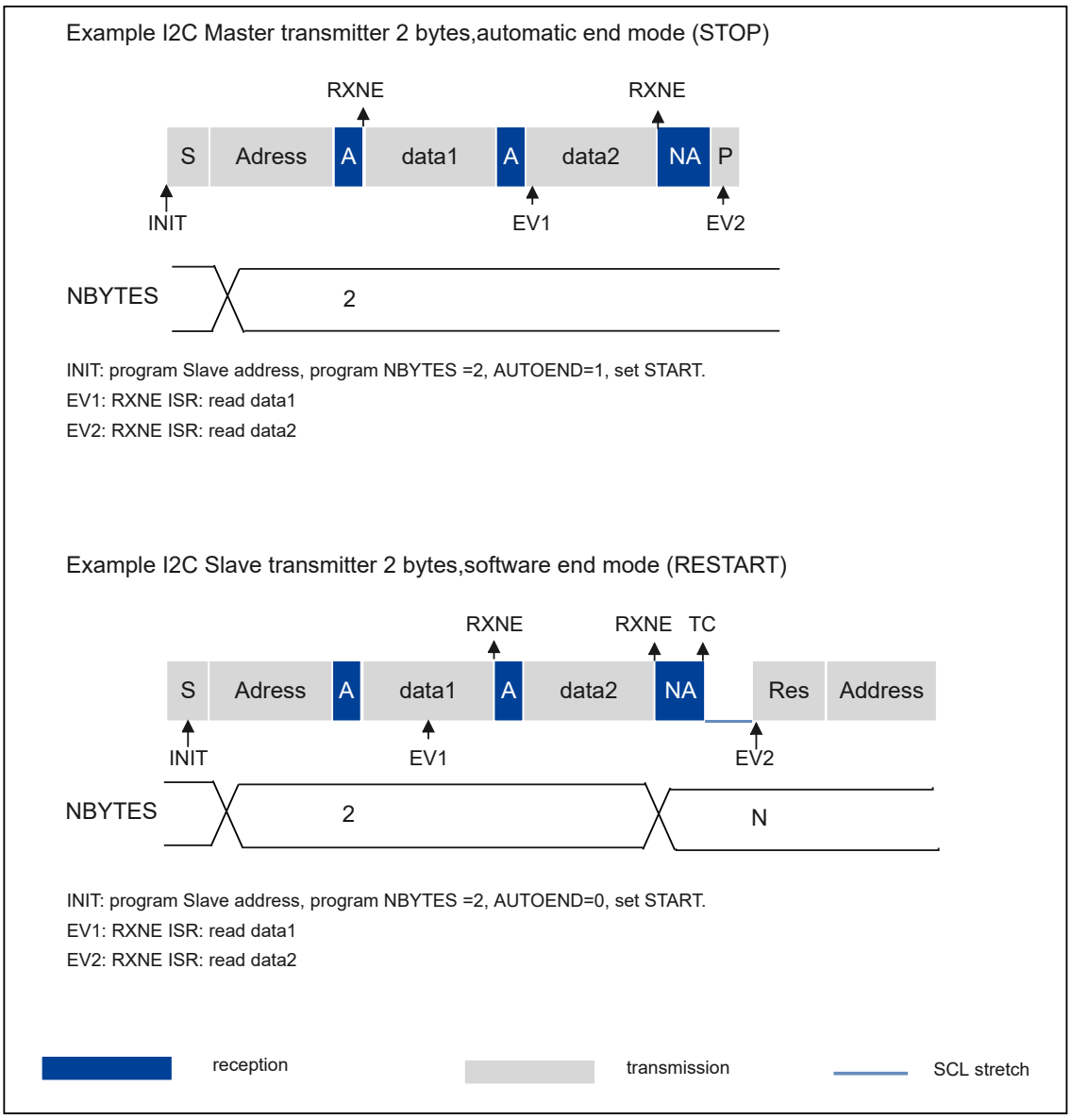


Figure 24-23 Transfer Bus Diagrams for I2C Master Receiver

24.3.10 I2C_TIMINGR Register Configuration Example

The tables below provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I2C specification.

Table 24-7 Examples of Timing Settings for f_{I2CCLK} = 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PREDIV	1	1	0	0
SCLLOW	0xC7	0x13	0x9	0x6
t_{SCLLOW}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLHI	0xC3	0xF	0x3	0x3

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
t_{SCLHI}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~2000 ns(4)
SDADEL	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

(1) SCL period t_{SCL} is greater than t_{SCLLOW} + t_{SCLHI} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.

(2) t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.

(3) t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.

(4) t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 24-8 Examples of Timings Settings for f_{I2CCLK} = 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PREDIV	3	3	1	0
SCLLOW	0xC7	0x13	0x9	0x4
t_{SCLLOW}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLHI	0xC3	0xF	0x3	0x2
t_{SCLHI}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~1000 ns(4)
SDADEL	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

(1) SCL period t_{SCL} is greater than t_{SCLLOW} + t_{SCLHI} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.

(2) t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.

(3) t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.

(4) t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

Table 24-9 Examples of Timings Settings for f_{I2CCLK} = 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PREDIV	0xB	0xB	5	5
SCLLOW	0xC7	0x13	0x9	0x3
t_{SCLLOW}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLHI	0xC3	0xF	0x3	0x1
t_{SCLHI}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
t_{SCL} (1)	~100 µs(2)	~10 µs(2)	~2500 ns(3)	~875 ns(4)
SDADEL	0x2	0x2	0x3	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

(1) The SCL period t_{SCL} is greater than t_{SCLLOW} + t_{SCLHI} due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.

(2) t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.

(3) t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.

(4) t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 250 ns.

24.3.11 SMBus Support

NOTE: This section is relevant only when the System Management Bus (SMBus) feature is supported.

The SMBus is a two-wire interface through which various devices can communicate with each other and the rest of the system. It is based on I2C principles of operation. The SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the http://smbus.org. The SMBus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as a master or slave device, and also as a host.

24.3.11.1 Bus Protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols 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. These protocols should be implemented by the user software.

For more details of these protocols, refer to http://smbus.org.

24.3.11.2 Address Resolution Protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. To provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of these protocols, refer to http://smbus.org.

24.3.11.3 Received Command and Data Acknowledge Control

A SMBus receiver must be able to NACK each received command or data. To allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to Slave Byte Control Mode for more details.

24.3.11.4 Host Notify Protocol

This peripheral supports the host notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

24.3.11.5 SMBus Alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the alert response address.

When configured as a slave device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN = 1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN = 0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN = 0.

24.3.11.6 Packet Error Checking

A packet error checking mechanism has been introduced in the SMBus Specification to improve reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the CRC-8 polynomial C(x) = x8 + x2 + x + 1 on all the message bytes, including addresses and read/write bits.

The peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

24.3.11.7 Timeouts

This peripheral embeds hardware timers to be compliant with the 3 timeouts defined in SMBus Specification.

Table 24-10 SMBus Timeout Specifications

Symbol	Parameter	Limits		Unit
		Min.	Max.	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (slave device)	-	25	ms
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (master device)	-	10	ms

(1) $t_{LOW:SEXT}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another slave device or the master also extends the clock, causing the combined clock low extend time to exceed $t_{LOW:SEXT}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.

(2) $t_{LOW:MEXT}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

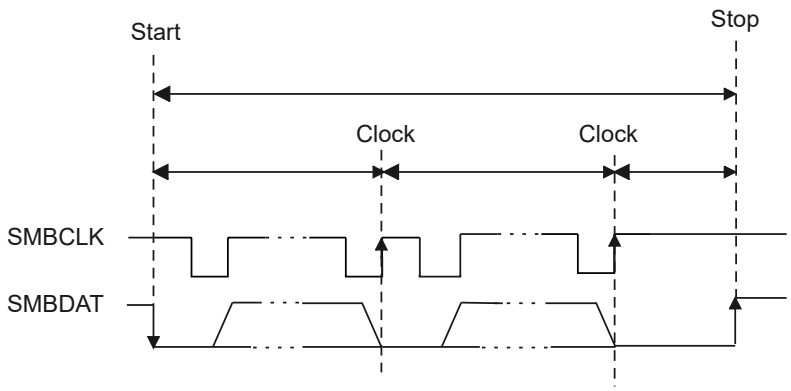


Figure 24-24 Timeout Intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$

24.3.11.8 Bus Idle Detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for t_{IDLE} greater than $t_{HIGH,MAX}$ (refer to Table 24-4).

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

24.3.12 SMBus Initialization

NOTE: This section is relevant only when SMBus feature is supported.

In addition to I2C initialization, some other specific initialization must be done to perform SMBus communication.

24.3.12.1 Received Command and Data Acknowledge Control (Slave Mode)

A SMBus receiver must be able to NACK each received command or data. To allow ACK control in slave mode, the Slave byte control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to Slave Byte Control Mode for more details.

24.3.12.2 Specific Address (Slave Mode)

The specific SMBus addresses must be enabled if needed. Refer to Bus Idle Detection for more details.

- The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The alert response address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.

24.3.12.3 Packet Error Checking

Packet error checking calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then, the PEC transfer is managed with the help of a hardware byte counter: NBYTE[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTE-1 data has been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

NOTE: Changing the PECEN configuration is not allowed when the I2C is enabled.

Table 24-11 SMBus with PEC Configuration

Mode	SBC Bit	RELOAD Bit	AUTOEND Bit	PECBYTE Bit
Master Tx/Rx NBYTE + PEC + STOP	x	0	1	1
Master Tx/Rx NBYTE + PEC + ReSTART	x	0	0	1
Slave Tx/Rx with PEC	1	0	x	1

24.3.12.4 Timeout Detection

The timeout detection is enabled by setting the TOEN and EXTTOEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus Specification.

- $t_{TIMEOUT}$ check

To enable the $t_{TIMEOUT}$ check, the 12-bit TOA[11:0] bits must be programmed with the timer reload value to check the $t_{TIMEOUT}$ parameter. The TOIDLE bit must be configured to '0' to detect the SCL low level timeout.

Then the timer is enabled by setting the TOEN in the I2C_TIMEOUTR register.

If SCL is tied low for a time greater than (TOA+1) x 2048 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register. Refer to Table 24-12.

NOTE: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

Depending on if the peripheral is configured as a master or as a slave, The 12-bit TOB timer must be configured to check $t_{LOW:SEXT}$ for a slave and $t_{LOW:MEXT}$ for a master. As the standard specifies only a maximum, the user can choose the same value for the both.

Then the timer is enabled by setting the EXTTOEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TOB+1) x 2048 x t_{I2CCLK}, and in the timeout interval described in Bus Idle Detection, the TIMEOUT flag is set in the I2C_ISR register.

Refer to Table 24-13.

NOTE: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

24.3.12.5 Bus Idle Detection

To enable the t_{IDLE} check, the 12-bit TOA[11:0] field must be programmed with the timer reload value to obtain the t_{IDLE} parameter. The TOIDLE bit must be configured to 1 to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TOEN bit in the I2C_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than (TOA+1) x 4 x t_{I2CCLK}, the TIMEOUT flag is set in the I2C_ISR register.

Refer to Table 24-14.

NOTE: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

24.3.13 SMBus: I2C_TIMEOUTR Register Configuration Examples

This section is relevant only when SMBus feature is supported.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 24-12 TOA Settings for Various I2CCLK Frequencies (Max $t_{TIMEOUT}$ = 25 ms) Examples

f_{I2CCLK}	TOA[11:0] Bits	TOIDLE Bit	TIMEOUTEN Bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
32 MHz	0x186	0	1	391 x 2048 x 31.25 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

- Configuring the maximum duration of $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ to 8 ms:

Table 24-13 TOB Settings for Various I2CCLK Frequencies Examples

f_{I2CCLK}	TOB[11:0] Bits	EXTTOEN Bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms

f_{I2CCLK}	TOB[11:0] Bits	EXTTOEN Bit	$t_{LOW:EXT}$
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 µs:

Table 24-14 TOA Settings for Various I2CCLK Frequencies (Max t_{IDLE} = 50 µs) Examples

f_{I2CCLK}	TOA[11:0] Bits	TOIDLE Bit	TIMEOUTEN Bit	t_{TOIDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 µs

24.3.14 SMBus Mode

NOTE: This section is relevant only when the SMBus feature is supported.

In addition to I2C slave transfer management (refer to I2C Slave Mode), some additional software flowcharts are provided to support the SMBus.

24.3.14.1 SMBus Slave Transmitter

When the IP is used in SMBus, SBC must be programmed to ‘1’ to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTE[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTE-1 and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTE-1 data transfer.

NOTE: The PECBYTE bit has no effect when the RELOAD bit is set.

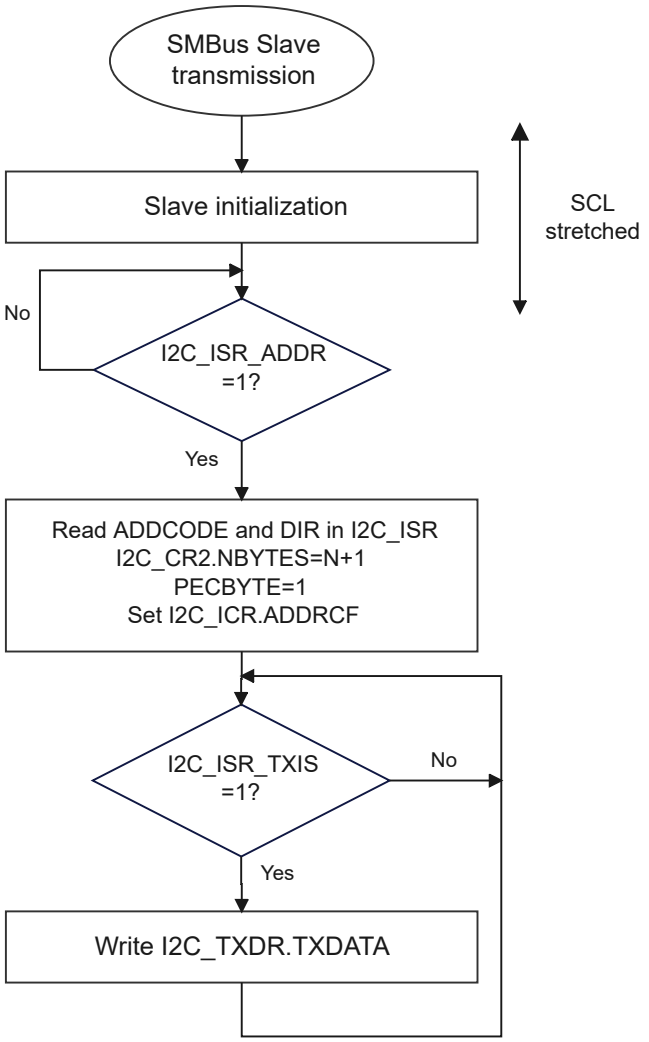


Figure 24-25 Transfer Sequence Flowchart for SMBus Slave Transmitter N Bytes + PEC

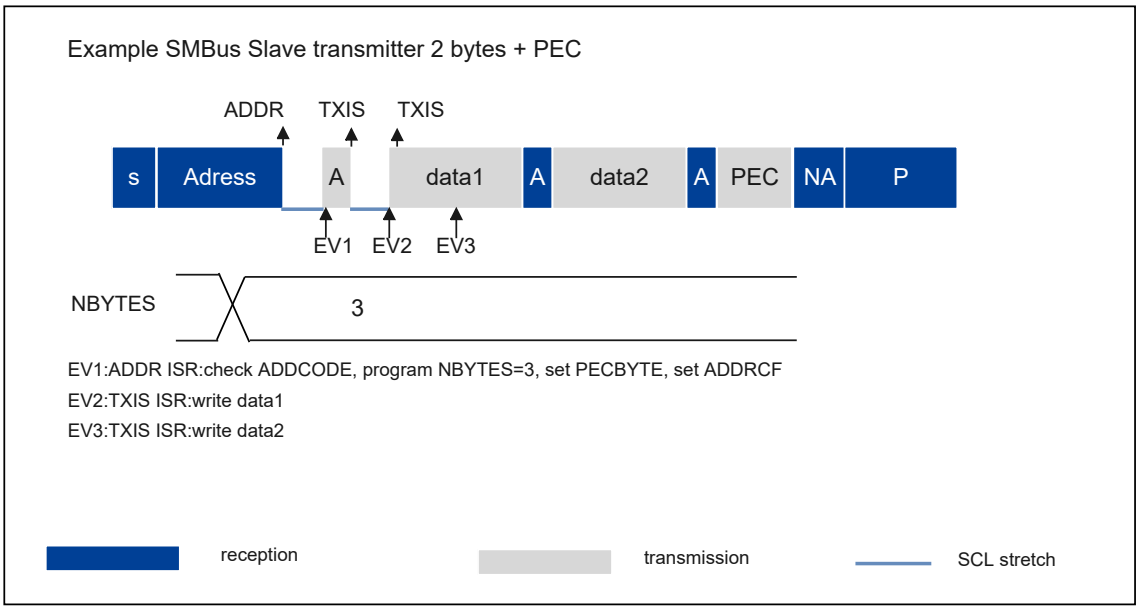


Figure 24-26 Transfer Bus Diagrams for SMBus Slave Transmitter (SBC = 1)

24.3.14.2 SMBus Slave Receiver

When the I2C is used in SMBus mode, SBC must be programmed to '1' to allow the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to Slave Byte Control Mode for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTE-1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, the user can program PECBYTE = 1 and, in the same write operation, program NBYTE with the number of bytes to be received in a continuous flow. After NBYTE-1 are received, the next received byte is checked as being the PEC.

NOTE: The PECBYTE bit has no effect when the RELOAD bit is set.

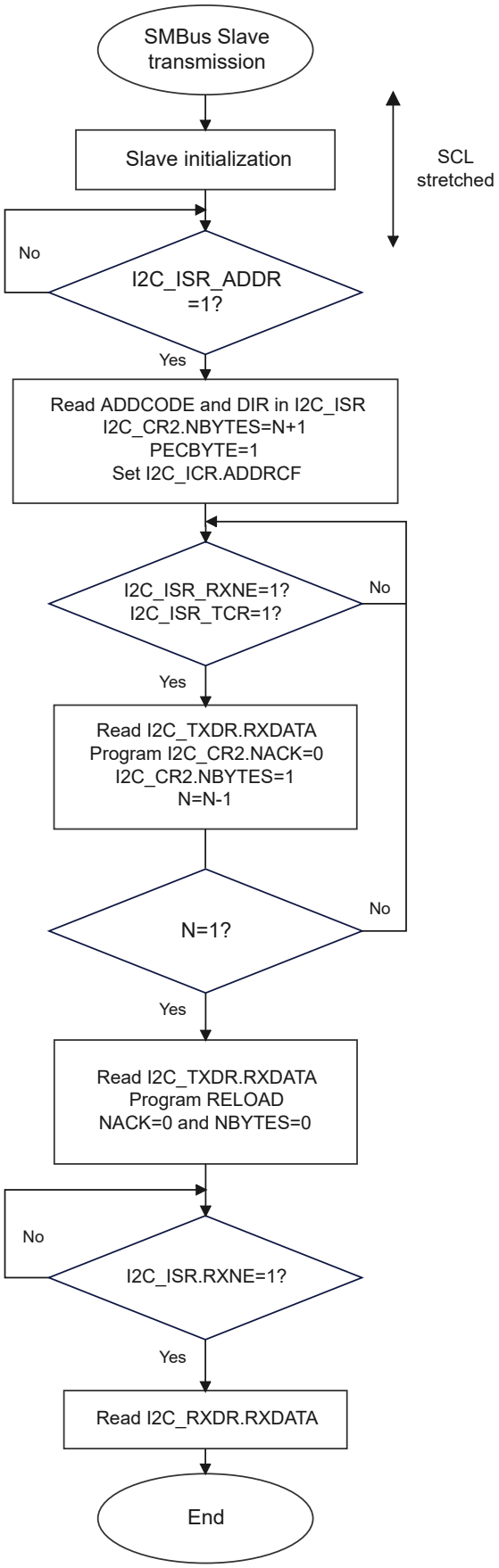


Figure 24-27 Transfer Sequence Flowchart for SMBus slave Receiver N Bytes + PEC

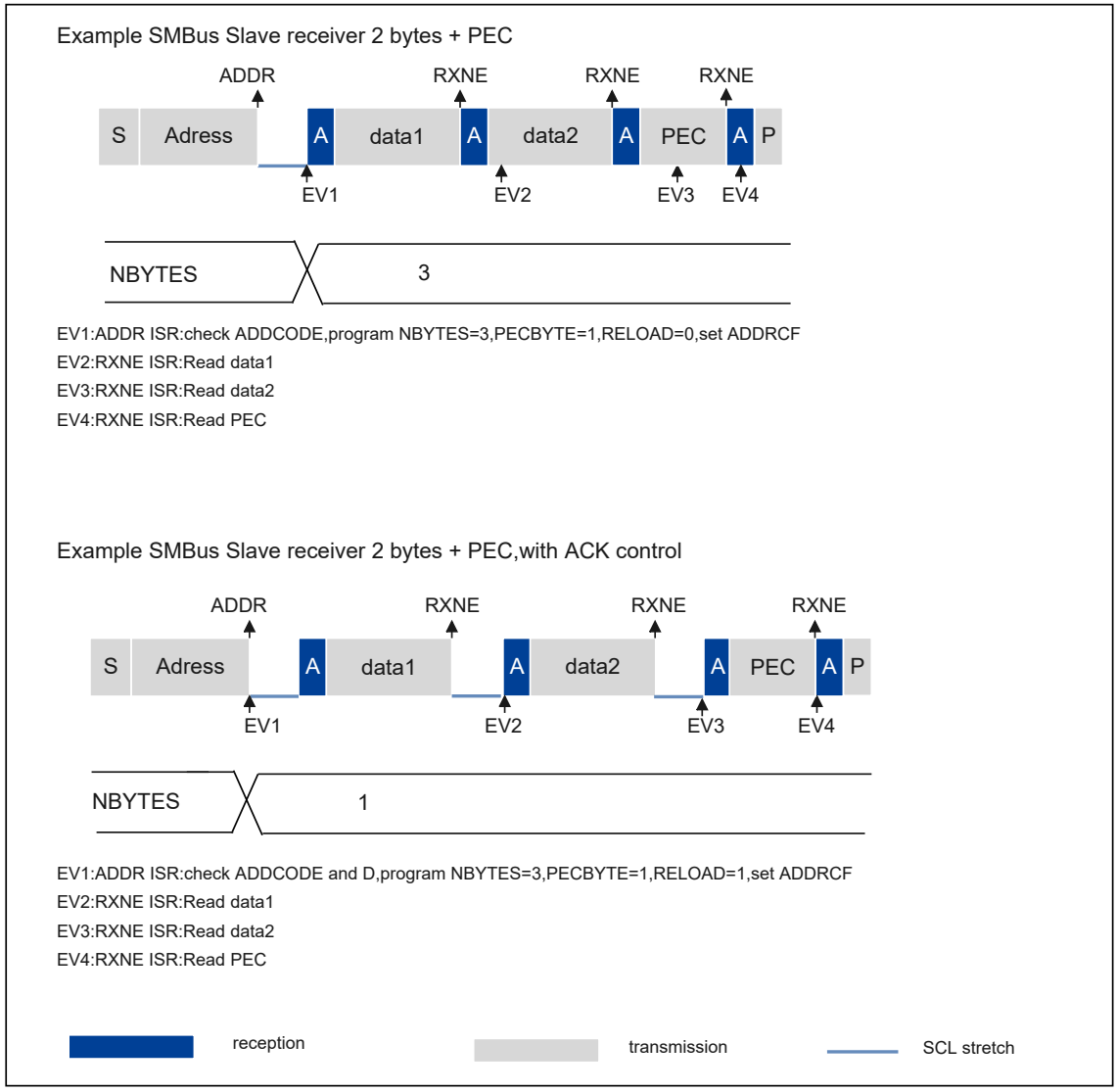


Figure 24-28 Bus Transfer Diagrams for SMBus Slave Receiver (SBC = 1)

24.3.14.3 SMBus Master Transmitter

When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTE[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTE-1. So if the PECBYTE bit is set when NBYTE = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND = 0). In this case, once NBYTE-1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

NOTE: The PECBYTE bit has no effect when the RELOAD bit is set.

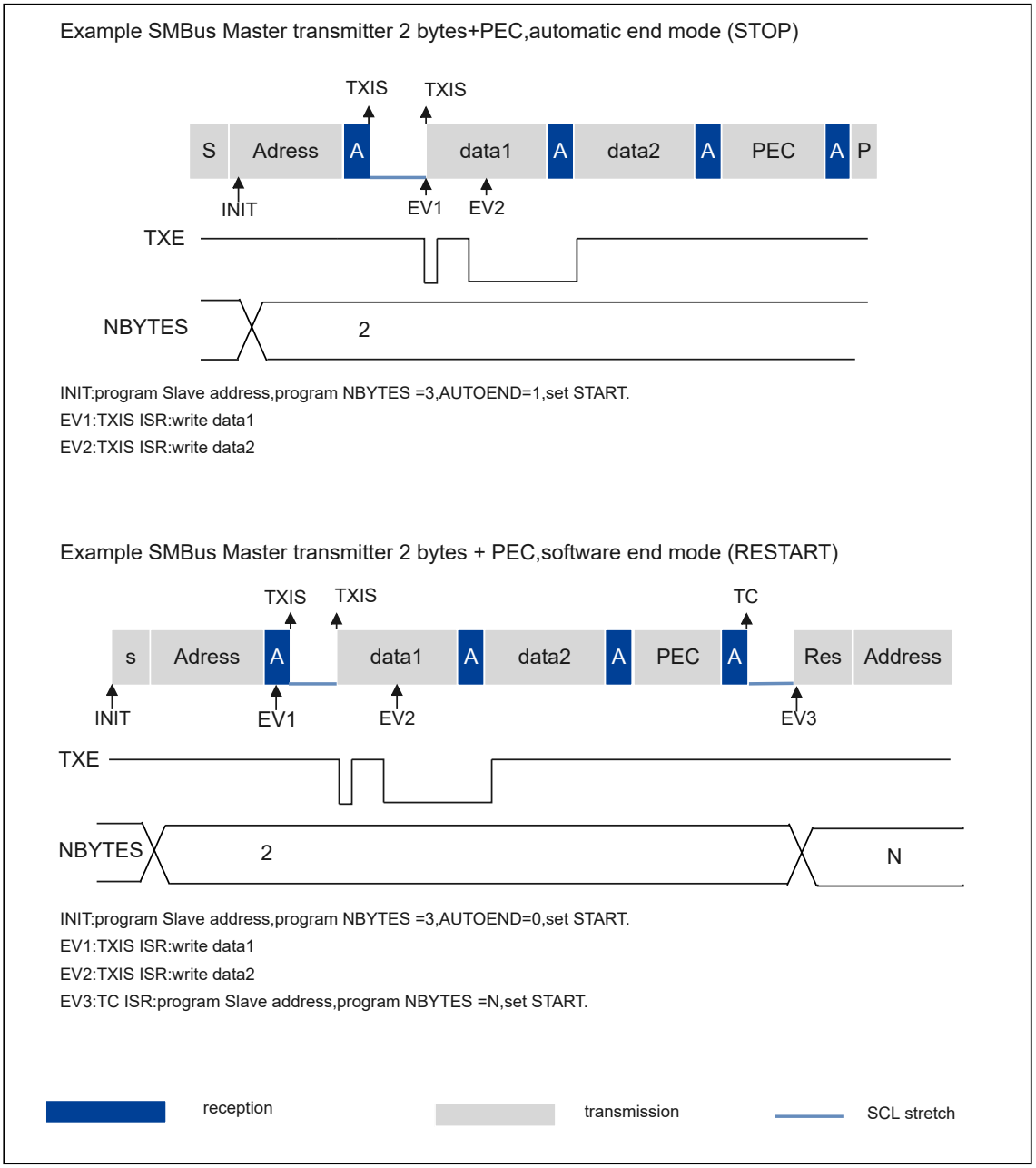


Figure 24-29 Bus Transfer Diagrams for SMBus Master Transmitter

24.3.14.4 SMBus Master Receiver

When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTE-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTE-1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

NOTE: The PECBYTE bit has no effect when the RELOAD bit is set.

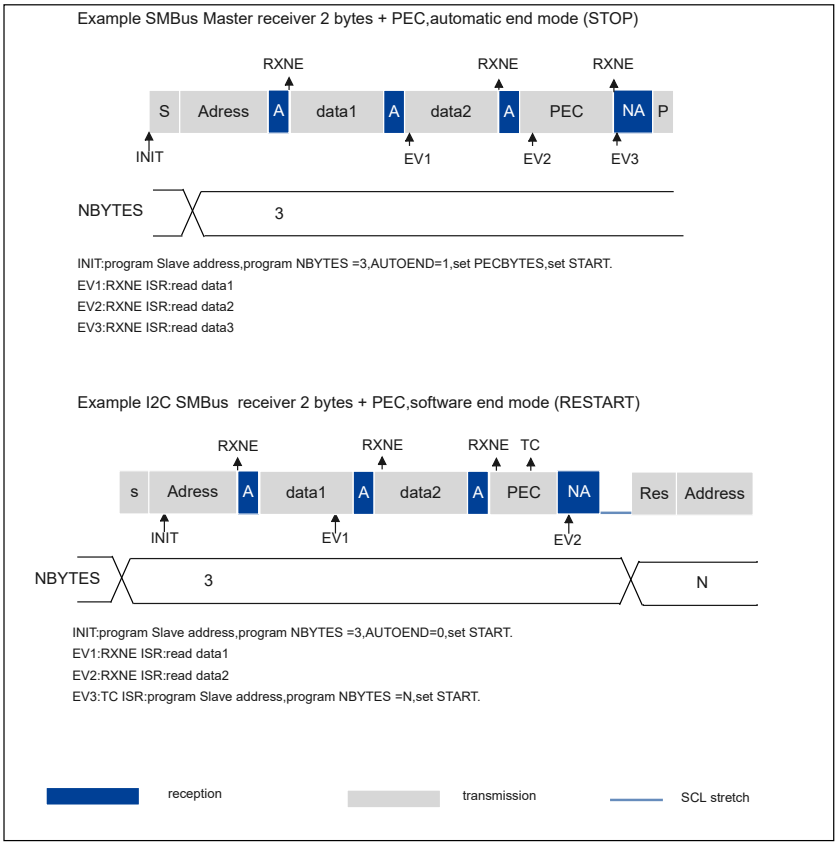


Figure 24-30 Bus Transfer Diagrams for SMBus Master Receiver

24.3.15 DMA Requests

24.3.15.1 Transmission Using DMA

Direct Memory Access (DMA) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see Direct memory access controller (DMA)) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTE counter. Refer to Master Transmitter.
- In slave mode:
  - With NOEXT = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOEXT = 1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTE counter. Refer to SMBus Slave Transmitter and SMBus Master Transmitter .

NOTE: If DMA is used for transmission, the TXIE bit does not need to be enabled.

24.3.15.2 Reception Using DMA

Direct Memory Access (DMA) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to Direct Memory Access (DMA)) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTE counter.

  In slave mode with NOEXT = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

  If SMBus is supported: the PEC transfer is managed with the NBYTE counter. Refer to SMBus Slave Receiver and SMBus Master Receiver.

NOTE: If DMA is used for reception, the RXIE bit does not need to be enabled.

24.3.16 Debug Mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_ configuration bits in the DBG module.

24.4 Errors and Interrupts

Table 24-15 lists the I2C error flags.

Table 24-15 I2C Error Flags

Error Acronym	Error Name	Description
BERR	Bus Error	A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when an SDA edge occurs while SCL is high. The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e. not during the address phase in slave mode). In case of a misplaced START or RESTART detection in slave mode, the I2C enters an address recognition state like for a correct START condition. When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
ARLO	Arbitration Lost	An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge. • In master mode, arbitration loss is detected during the address phase, data phase, and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware, and the master switches automatically to slave mode. • In slave mode, arbitration loss is detected during the data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released. When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
OVR	Overrun/Underrun Error	An overrun or underrun error is detected in slave mode when NOEXT = 1 and: • In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte. • In transmission:

Error Acronym	Error Name	Description
		– When STOPF = 1 and the first data byte should be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not. – When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent. When an overrun or underrun error is detected, the OVF flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
PECERR	Packet Error Checking Error	**NOTE:** This part is relevant only when the SMBus feature is supported. A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception. When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
TIMEOUT	Timeout Error	**NOTE:** This part is relevant only when the SMBus feature is supported. A timeout error occurs for any of these conditions: • TOIDLE = 0 and SCL remained low for the time defined in the TOA[11:0] bits: this is used to detect an SMBus timeout. • TOIDLE = 1 and both SDA and SCL remained high for the time defined in the TOA[11:0] bits: this is used to detect a bus idle condition. • Master cumulative clock low extend time reached the time defined in the TOB[11:0] bits (SMBus $t_{LOW:MEXT}$ parameter) • Slave cumulative clock low extend time reached the time defined in TOB[11:0] bits (SMBus $t_{LOW:SEXT}$ parameter) When a timeout violation is detected in master mode, a STOP condition is automatically sent. When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released. When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Error Acronym	Error Name	Description
ALERT	Alert	**NOTE:** This part is relevant only when the SMBus feature is supported. The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN = 1), the alert pin detection is enabled (ALERTEN = 1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Table 24-16 lists the I2C interrupt requests.

Table 24-16 I2C Interrupt Requests

Interrupt Acronym		Interrupt Event	Event Flag	Enable Control Bit	Interrupt Clear Method	Exit the Sleep Mode	Exit the Stop Mode	Exit the Standby Mode
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	No
		Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
		Stop detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
		Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTE[7:0] ≠ 0			
		Transfer complete	TC		Write START = 1 or STOP = 1			
		Address matched	ADDR	ADDRIE	Write ADDRCR = 1		No	
		NACK reception	NACKF	NACKIE	Write NACKCF = 1		No	
	I2C_ER	Bus error	BERR	ERRIE	Write BERRCF = 1	Yes	No	No
		Arbitration loss	ARLO		Write ARLOCF = 1			
		Overrun/ Underrun	OVF		Write OVFCF = 1			

Interrupt Acronym		Interrupt Event	Event Flag	Enable Control Bit	Interrupt Clear Method	Exit the Sleep Mode	Exit the Stop Mode	Exit the Standby Mode
		PEC error	PECERR		Write PECERRCF = 1			
		Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF = 1			
		SMBus alert	ALERT		Write ALERTCF = 1			

24.5 Registers

24.5.1 Register Address Map

Offset	Register Name	Register Description
0x0000	I2C_CR1	I2C control register 1
0x0004	I2C_CR2	I2C control register 2
0x0008	I2C_OAR1	I2C own address 1 register
0x000c	I2C_OAR2	I2C own address 2 register
0x0010	I2C_TIMINGR	I2C timing register
0x0014	I2C_TIMEOUTR	I2C timeout register
0x0018	I2C_ISR	I2C interrupt and status register
0x001c	I2C_ICR	I2C interrupt clear register
0x0020	I2C_PECR	I2C PEC register
0x0024	I2C_RXDR	I2C receive data register
0x0028	I2C_TXDR	I2C transmit data register

24.5.2 Register Field Details

24.5.2.1 I2C_CR1

0x0000			I2C control register 1											I2C_CR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOEXT	SBCEN
Type	RO								WR	WR	WR	WR	WR	WR	WR	WR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RXDMAEN	TXDMAEN	Reserved	ANFOFF	DNF				ERRIEN	TCIEN	STOPIEN	NACKIEN	ADDRIEN	RXIEN	TXIEN	I2CEN
Type	WR	WR	RO	WR	WR				WR	WR	WR	WR	WR	WR	WR	WR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-17 I2C Control Register 1 Description

Field	Name	Description
31:24	Reserved	Reserved
23	PECEN	PEC enable 0: PEC calculation disabled 1: PEC calculation enabled
22	ALERTEN	SMBus alert enable 0: The SMBus alert pin (SMBA) is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released, and the Alert Response Address header is disabled (0001100x followed by NACK). 1: The SMBus alert pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low, and the Alert Response Address header is enabled (0001100x followed by ACK).

Field	Name	Description
		NOTE: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.
21	SMBDEN	SMBus device default address enable 0: Device default address disabled. Address 0b1100001x is NACKed. 1: Device default address enabled. Address 0b1100001x is ACKed.
20	SMBHEN	SMBus host address enable 0: Host address disabled. Address 0b0001000x is NACKed. 1: Host address enabled. Address 0b0001000x is ACKed.
19	GCEN	General call enable 0: General call disabled. Address 0b00000000 is NACKed. 1: General call enabled. Address 0b00000000 is ACKed.
18	WUPEN	Wakeup from stop mode enable 0: Wakeup from stop mode disabled 1: Wakeup from stop mode enabled
17	NOEXT	Clock stretching disable This bit is used to disable clock stretching in slave mode. It must be kept clear in master mode. 0: Clock stretching enabled 1: Clock stretching disabled
16	SBCEN	Slave byte control This bit is used to enable hardware byte control in slave mode. 0: Slave byte control disabled 1: Slave byte control enabled
15	RXDMAEN	DMA reception requests enable 0: DMA mode disabled for reception 1: DMA mode enabled for reception

Field	Name	Description
14	TXDMAEN	DMA transmission requests enable 0: DMA mode disabled for transmission 1: DMA mode enabled for transmission
13	Reserved	Reserved
12	ANFOFF	Analog noise filter OFF 0: Analog noise filter enabled 1: Analog noise filter disabled **NOTE:** This bit can only be programmed when the I2C is disabled (I2CEN = 0).
11:8	DNF	Digital noise filter These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter filters spike with a length up to DNF[3:0] * $t_{I2CCLK.}$ 0000: Digital filter disabled 0001: Digital filter enabled and filtering capability up to 1 t_{I2CCLK} ... 1111: Digital filter enabled and filtering capability up to 15 t_{I2CCLK}
7	ERRIEN	Error interrupts enable 0: Error detection interrupts disabled 1: Error detection interrupts enabled **NOTE:** Any of these errors generate an interrupt: • Arbitration Loss (ARLO) • Bus Error detection (BERR) • Overrun/Underrun (OVR) • PEC error detection (PECERR) • Alert pin event detection (ALERT)

Field	Name	Description
		• Timeout detection (TIMEOUT)
6	TCIEN	Transfer Complete interrupt enable 0: Transfer Complete interrupt disabled 1: Transfer Complete interrupt enabled Any of these events generate an interrupt: **NOTE:** • Transfer Complete (TC) • Transfer Complete Reload (TCR)
5	STOPIEN	Stop detection Interrupt enable 0: Stop detection (STOPF) interrupt disabled 1: Stop detection (STOPF) interrupt enabled
4	NACKIEN	Not acknowledge received Interrupt enable 0: Not acknowledge (NACKF) received interrupts disabled 1: Not acknowledge (NACKF) received interrupts enabled
3	ADDRIEN	Address match Interrupt enable (slave only) 0: Address match (ADDR) interrupts disabled 1: Address match (ADDR) interrupts enabled
2	RXIEN	RX Interrupt enable 0: Receive (RXNE) interrupt disabled 1: Receive (RXNE) interrupt enabled
1	TXIEN	TX Interrupt enable 0: Transmit (TXIS) interrupt disabled 1: Transmit (TXIS) interrupt enabled
0	I2CEN	I2C enable

Field	Name	Description
		0: Peripheral I2C disabled 1: Peripheral I2C enabled **NOTE:** When I2CEN = 0, the I2C SCL, and SDA lines are released. Internal state machines and status bits are put back to their reset value. I2CEN must be kept low for at least 3 APB clock cycles when cleared.

24.5.2.2 I2C_CR2

0x0004			I2C control register 2											I2C_CR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved					PECBYTE	AUTOEND	RELOAD	NBYTE							
Type	RO					WR	WR	WR	WR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	NACK	STOP	START	Reserved	ADD10	RDWRN	SADD									
Type	WR	WR	WR	RO	WR	WR	WR									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-18 I2C Control Register 2 Description

Field	Name	Description
31:27	Reserved	Reserved
26	PECBYTE	Packet error checking byte This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when I2CEN = 0. 0: No PEC transfer 1: PEC transmission/reception is requested.
25	AUTOEND	Automatic end mode (master mode) This bit is set and cleared by software. 0: Software end mode. TC flag is set when NBYTES data are transferred, stretching SCL low. 1: Automatic end mode. A STOP condition is automatically sent when NBYTES data are transferred.
24	RELOAD	NBYTES reload mode

Field	Name	Description
		This bit is set and cleared by software. 0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows). 1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.
23:16	NBYTE	Number of bytes The number of bytes to be transmitted/received is programmed there. This field doesn't care in slave mode with SBC = 0.
15	NACK	NACK generation (slave mode) The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when I2CEN = 0. 0: An ACK is sent after current received byte. 1: A NACK is sent after current received byte.
14	STOP	Stop generation (master mode) The bit is set by software, cleared by hardware when a STOP condition is detected, or when I2CEN = 0. In master mode: 0: No Stop generation 1: Stop generation after current byte transfer **NOTE:** Writing '0' to this bit has no effect.
13	START	Start generation This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when I2CEN = 0. It can also be cleared by software by writing '1' to the ADDRCR bit in the I2C_ICR register. 0: No Start generation 1: Restart/Start generation

Field	Name	Description
		If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD = 0, after the end of the NBYTES transfer. Otherwise, setting this bit generates a START condition once the bus is free. **NOTE:** Writing '0' to this bit has no effect. The START bit can be set even if the bus is BUSY or I2C is in slave mode. This bit has no effect when RELOAD is set.
12	Reserved	Reserved
11	ADD10	10-bit addressing mode (master mode) 0: The master operates in 7-bit addressing mode. 1: The master operates in 10-bit addressing mode. **NOTE:** Changing this bit when the START bit is set is not allowed.
10	RDWRN	Transfer direction (master mode) 0: Master requests a write transfer. 1: Master requests a read transfer. **NOTE:** Changing this bit when the START bit is set is not allowed.
9:0	SADD	In 7-bit addressing mode (ADD10 = 0): SADD[7:1] should be written with the 7-bit slave address to be sent. The bits SADD[9], SADD[8] and SADD[0] are don't care. In 10-bit addressing mode (ADD10 = 1): SADD[9:0] should be written with the 10-bit slave address to be sent. **NOTE:** Changing these bits when the START bit is set is not allowed.

24.5.2.3 I2C_OAR1

0x0008			I2C own address 1 register											I2C_OAR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OA1EN	Reserved				OA1MODE	OA198		OA171							OA10
Type	WR	RO				WR	WR		WR							WR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-19 I2C Own Address 1 Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	OA1EN	Own Address 1 enable 0: Own address 1 disabled. The received slave address OA1 is NACKed. 1: Own address 1 enabled. The received slave address OA1 is ACKed.
14:11	Reserved	Reserved
10	OA1MODE	Own Address 1 10-bit mode 0: Own address 1 is a 7-bit address. 1: Own address 1 is a 10-bit address. **NOTE:** This bit can be written only when OA1EN = 0.
9:8	OA198	Interface own slave address

Field	Name	Description
		10-bit mode: 10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address. 7-bit mode: The bits OA1[9], OA1[8], and OA1[0] don't care. 10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.
7:1	OA171	Interface own slave address 10-bit mode: 10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address. 7-bit mode: The bits OA1[9], OA1[8] and OA1[0] are don't care. 10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.
0	OA10	Interface own slave address 10-bit mode: 10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address. 7-bit mode: The bits OA1[9], OA1[8], and OA1[0] don't care. 10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.

24.5.2.4 I2C_OAR2

0x000c			I2C own address 2 register											I2C_OAR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OA2EN	Reserved				OA2MSK			OA271							Reserved
Type	WR	RO				WR			WR							RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-20 I2C Own Address 2 Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	OA2EN	Own Address 2 enable 0: Own address 2 disabled. The received slave address OA2 is NACKed. 1: Own address 2 enabled. The received slave address OA2 is ACKed.
14:11	Reserved	Reserved
10:8	OA2MSK	Own Address 2 masks 000: No mask 001: OA2[1] is masked and doesn't care. Only OA2[7:2] is compared. 010: OA2[2:1] are masked and don't care. Only OA2[7:3] is compared. 011: OA2[3:1] are masked and don't care. Only OA2[7:4] is compared. 100: OA2[4:1] are masked and don't care. Only OA2[7:5] is compared.

Field	Name	Description
		101: OA2[5:1] are masked and don't care. Only OA2[7:6] is compared. 110: OA2[6:1] are masked and don't care. Only OA2[7] is compared. 111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged. **NOTE:** These bits can be written only when OA2EN = 0. As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.
7:1	OA271	Interface address 7-bit addressing mode: 7-bit address **NOTE:** These bits can be written only when OA2EN = 0.
0	Reserved	Reserved

24.5.2.5 I2C_TIMINGR

0x0010			I2C timing register											I2C_TIMINGR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PREDIV				Reserved				SCLDEL				SDADEL			
Type	WR				RO				WR				WR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SCLHI								SCLLOW							
Type	WR								WR							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-21 I2C Timing Register Description

Field	Name	Description
31:28	PREDIV	Timing prescaler This field is used to prescale I2CCLK to generate the clock period t_{PREDIV} used for data setup and hold counters and for SCL high and low level counters. t_{PREDIV} = (PREDIV+1) x t_{I2CCLK}
27:24	Reserved	Reserved
23:20	SCLDEL	Data setup time This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge. In master mode and in slave mode with NOEXT = 0, the SCL line is stretched low during t_{SCLDEL}. t_{SCLDEL} = (SCLDEL+1) x t_{PREDIV} **NOTE:** t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.
19:16	SDADEL	Data hold time

Field	Name	Description
		This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In master mode and in slave mode with NOEXT = 0, the SCL line is stretched low during $t_{SDADEL.}$ t_{SDADEL}= SDADEL x t_{PREDIV} **NOTE:** SDADEL is used to generate $t_{HD:DAT}$ timing.
15:8	SCLHI	SCL high period (master mode) This field is used to generate the SCL high period in master mode. t_{SCLHI} = (SCLHI+1) x t_{PREDIV} **NOTE:** SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.
7:0	SCLLOW	SCL low period (master mode) This field is used to generate the SCL low period in master mode. t_{SCLLOW} = (SCLLOW+1) x t_{PREDIV} **NOTE:** SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

24.5.2.6 I2C_TIMEOUTR

0x0014			I2C timeout register											I2C_TIMEOUTR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EXTTOEN	Reserved			TOB											
Type	WR	RO			WR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TOEN	Reserved		TOIDLE	TOA											
Type	WR	RO		WR	WR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-22 I2C Timeout Register Description

Field	Name	Description
31	EXTTOEN	Extended clock timeout enable 0: Extended clock timeout detection is disabled 1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).
30:28	Reserved	Reserved
27:16	TOB	Bus timeout B This field is used to configure the cumulative clock extension timeout: In master mode, the master cumulative clock low extend time ($t_{LOW:MEXT}$) is detected In slave mode, the slave cumulative clock low extend time ($t_{LOW:SEXT}$) is detected $t_{LOW:EXT}$ = (TOB + 1) x 2048 x t_{I2CCLK} **NOTE:** These bits can be written only when TEXTEN = 0.

Field	Name	Description
15	TOEN	Clock timeout enable 0: SCL timeout detection is disabled 1: SCL timeout detection is enabled When SCL is low for more than $t_{TIMEOUT}$ (TOIDLE = 0) or high for more than t_{IDLE} (TOIDLE = 1), a timeout error is detected (TIMEOUT = 1).
14:13	Reserved	Reserved
12	TOIDLE	Idle clock timeout detection 0: TOA is used to detect SCL low timeout 1: TOA is used to detect both SCL and SDA high timeout (bus idle condition) **NOTE:** This bit can be written only when TIMOUTEN = 0.
11:0	TOA	Bus Timeout A This field is used to configure: The SCL low timeout condition $t_{TIMEOUT}$ when TOIDLE = 0 $t_{TIMEOUT}$= (TOA+1) x 2048 x t_{I2CCLK} The bus idle condition (both SCL and SDA high) when TOIDLE = 1 t_{IDLE}= (TOA+1) x 4 x t_{I2CCLK} **NOTE:** These bits can be written only when TIMOUTEN = 0.

24.5.2.7 I2C_ISR

0x0018			I2C interrupt and status register											I2C_ISR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								ADDCODE							DIR
Type	RO								RO							RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BUSY	Reserved	ALERT	TIMEOUT	PECERR	OVF	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RS	RS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-23 I2C Interrupt and Status Register Description

Field	Name	Description
31:24	Reserved	Reserved
23:17	ADDCODE	Address match code (slave mode) These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.
16	DIR	Transfer direction (slave mode) This flag is updated when an address match event occurs (ADDR = 1). 0: Write transfer, slave enters receiver mode. 1: Read transfer, slave enters transmitter mode.
15	BUSY	Bus busy This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected, or when I2CEN = 0.
14	Reserved	Reserved

Field	Name	Description
13	ALERT	SMBus alert This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit. **NOTE:** This bit is cleared by hardware when I2CEN = 0.
12	TIMEOUT	Timeout or t_{LOW} detection flag This flag is set by hardware when a timeout or extended clock timeout occurs. It is cleared by software by setting the TIMEOUTCF bit. **NOTE:** This bit is cleared by hardware when I2CEN = 0.
11	PECERR	PEC Error in reception This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit. **NOTE:** This bit is cleared by hardware when I2CEN = 0.
10	OVF	Overrun/Underrun (slave mode) This flag is set by hardware in slave mode with NOEXT = 1 when an overrun/underrun error occurs. It is cleared by software by setting the OVFCF bit. **NOTE:** This bit is cleared by hardware when I2CEN = 0.
9	ARLO	Arbitration lost This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit. **NOTE:** This bit is cleared by hardware when I2CEN = 0.

Field	Name	Description
8	BERR	Bus error This flag is set by hardware when a misplaced Start or STOP condition is detected, whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting the BERRCF bit. **NOTE:** This bit is cleared by hardware when I2CEN = 0.
7	TCR	Transfer Complete Reload This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value. **NOTE:** This bit is cleared by hardware when I2CEN = 0. This flag is only for master mode or slave mode when the SBC bit is set.
6	TC	Transfer Complete (master mode) This flag is set by hardware when RELOAD = 0, AUTOEND = 0, and NBYTES data have been transferred. It is cleared by software when the START or STOP bit is set.
5	STOPF	Stop detection flag This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer: • either as a master, provided that the STOP condition is generated by the peripheral. • or as a slave, provided that the peripheral has been addressed previously during this transfer. It is cleared by software by setting the STOPCF bit. **NOTE:** This bit is cleared by hardware when I2CEN = 0.
4	NACKF	Not Acknowledge received flag

Field	Name	Description
		This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit. **NOTE:** This bit is cleared by hardware when I2CEN = 0.
3	ADDR	Address matched (slave mode) This bit is set by hardware as soon as the received slave address matches one of the enabled slave addresses. It is cleared by software by setting ADDRCR bit. **NOTE:** This bit is cleared by hardware when I2CEN = 0.
2	RXNE	Receive data register not empty (receivers) This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read. **NOTE:** This bit is cleared by hardware when I2CEN = 0.
1	TXIS	Transmit interrupt status (transmitters) This bit is set by hardware when the I2C_TXDR register is empty, and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to bemsent is written in the I2C_TXDR register. This bit can be written to '1' by software when NOEXT = 1 only to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1). **NOTE:** This bit is cleared by hardware when I2CEN = 0.
0	TXE	Transmit data register empty (transmitters) This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register. This bit can be written to '1' by software to flush the transmit data register I2C_TXDR.

Field	Name	Description
		NOTE: This bit is set by hardware when I2CEN = 0.

24.5.2.8 I2C_ICR

0x001c			I2C interrupt clear register											I2C_ICR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		ALERTCF	TIMEOUTCF	PECCF	OVFCF	ARLOCF	BERRCF	Reserved		STOPCF	NACKCF	ADDRCR	Reserved		
Type	RO		WC	WC	WC	WC	WC	WC	RO		WC	WC	WC	RO		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-24 I2C Interrupt Clear Register Description

Field	Name	Description
31:14	Reserved	Reserved
13	ALERTCF	Alert flag clear Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.
12	TIMEOUTCF	Timeout detection flag clear Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.
11	PECCF	PEC Error flag clear Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.
10	OVFCF	Overrun/Underrun flag clear Writing 1 to this bit clears the OVF flag in the I2C_ISR register
9	ARLOCF	Arbitration lost flag clear Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Field	Name	Description
8	BERRCF	Bus error flag clear Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.
7:6	Reserved	Reserved
5	STOPCF	STOP detection flag clear Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.
4	NACKCF	Not Acknowledge flag clear Writing 1 to this bit clears the NACKF flag in I2C_ISR register.
3	ADDRCR	Address matched flag clear Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.
2:0	Reserved	Reserved

24.5.2.9 I2C_PECR

0x0020			I2C PEC register											I2C_PECR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								PEC							
Type	RO								RO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-25 I2C PEC Register Description

Field	Name	Description
31:8	Reserved	Reserved
7:0	PEC	Packet error checking register This field contains the internal PEC when PECEN = 1. The PEC is cleared by hardware when I2CEN = 0.

24.5.2.10 I2C_RXDR

0x0024			I2C receive data register											I2C_RXDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								RXDATA							
Type	RO								RO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-26 I2C Receive Data Register Description

Field	Name	Description
31:8	Reserved	Reserved
7:0	RXDATA	8-bit receive data Data byte received from the I2C bus

24.5.2.11 I2C_TXDR

0x0028			I2C transmit data register											I2C_TXDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved								TXDR							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 24-27 I2C Transmit Data Register Description

Field	Name	Description
31:8	Reserved	Reserved
7:0	TXDR	8-bit transmit data Data byte to be transmitted to the I2C bus

Universal Synchronous/Asynchronous Receiver/Transmitter (UART/ USART)

This chapter describes the details of the Universal Synchronous/Asynchronous Receiver/Transmitter (USART).

Topics: **Page**

25.1 Introduction

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) supports standard UART and full-duplex asynchronous communication. The baud rate generator can generate a very wide range of baud rates for users. It also supports synchronous master mode, half-duplex single-wire communications, Local Interconnection Network (LIN), modem operations (CTS/RTS), and multiprocessor communications.

High-speed data communications can be achieved by using the Direct Memory Access (DMA).

25.2 Features

Main Features

- Full-duplex asynchronous communication
- Standard Non-Return-to-Zero (NRZ) format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data
- Tx and Rx FIFO must be enabled/disabled by software together
- A common programmable transmit and receive baud ratel
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection programmable data word length (5, 6, 7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (0.5, 1, 1.5 or 2 stop bits)
- Synchronous master mode and clock output for synchronous communications
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: Wakeup from mute mode by idle line detection or address mark detection

Extended Features

- 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
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

- Support loopback mode

25.3 Functional Description

25.3.1 Block Diagram

Figure 25-1 shows the block diagram of the USART module.

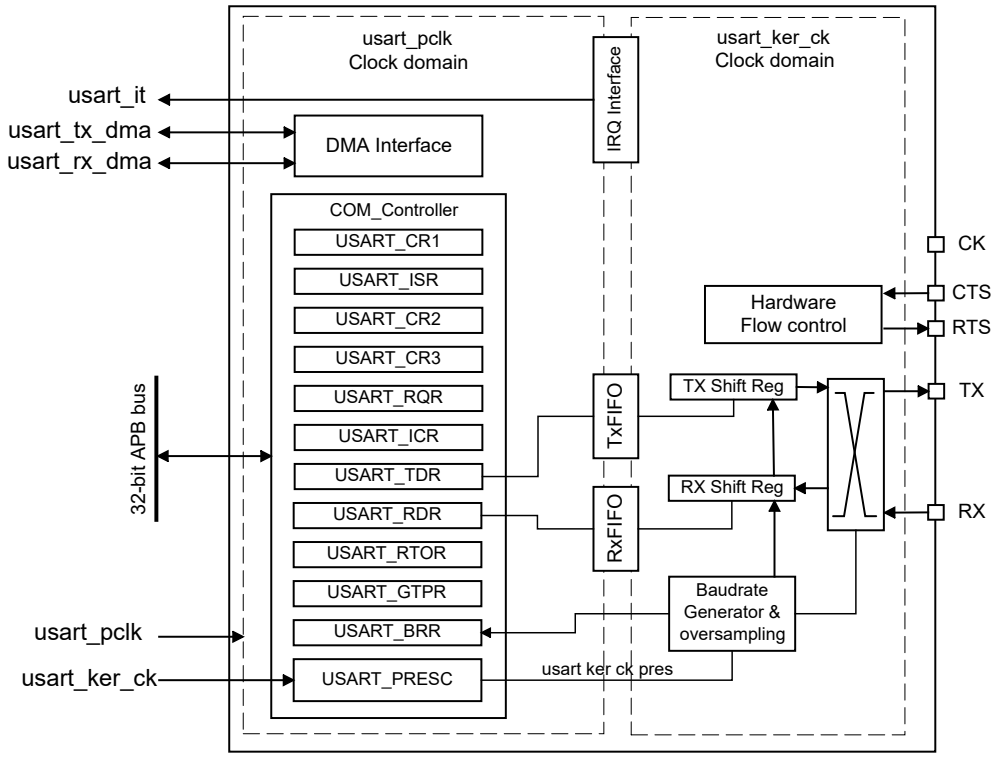


Figure 25-1 USART Block Diagram

The simplified block diagram given in Figure 25-1 shows two fully independent clock domains:

- Usart_pclk clock domain

  The usart_pclk clock signal feeds the peripheral bus interface.
- Usart_ker_ck kernel clock domain

  The usart_ker_ck is the USART clock source.

When the dual clock domain feature is disabled, the usart_ker_ck clock is the same as the usart_pclk clock.

25.3.2 Base Configuration

25.3.2.1 Character Length

The word length can be set to 5, 6, 7, 8 or 9 bits by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register, (M2: bit 2) in the USART_CR2 register:

M[2:0]	Character Length (Bit)
000	8

M[2:0]	Character Length (Bit)
001	9
010	7
100	5
101	6

NOTE: In 5-bit, 6-bit, or 7-bit data length mode, the LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

An Idle character is interpreted as an entire frame of “1”s (the number of “1”s includes 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 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock is generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

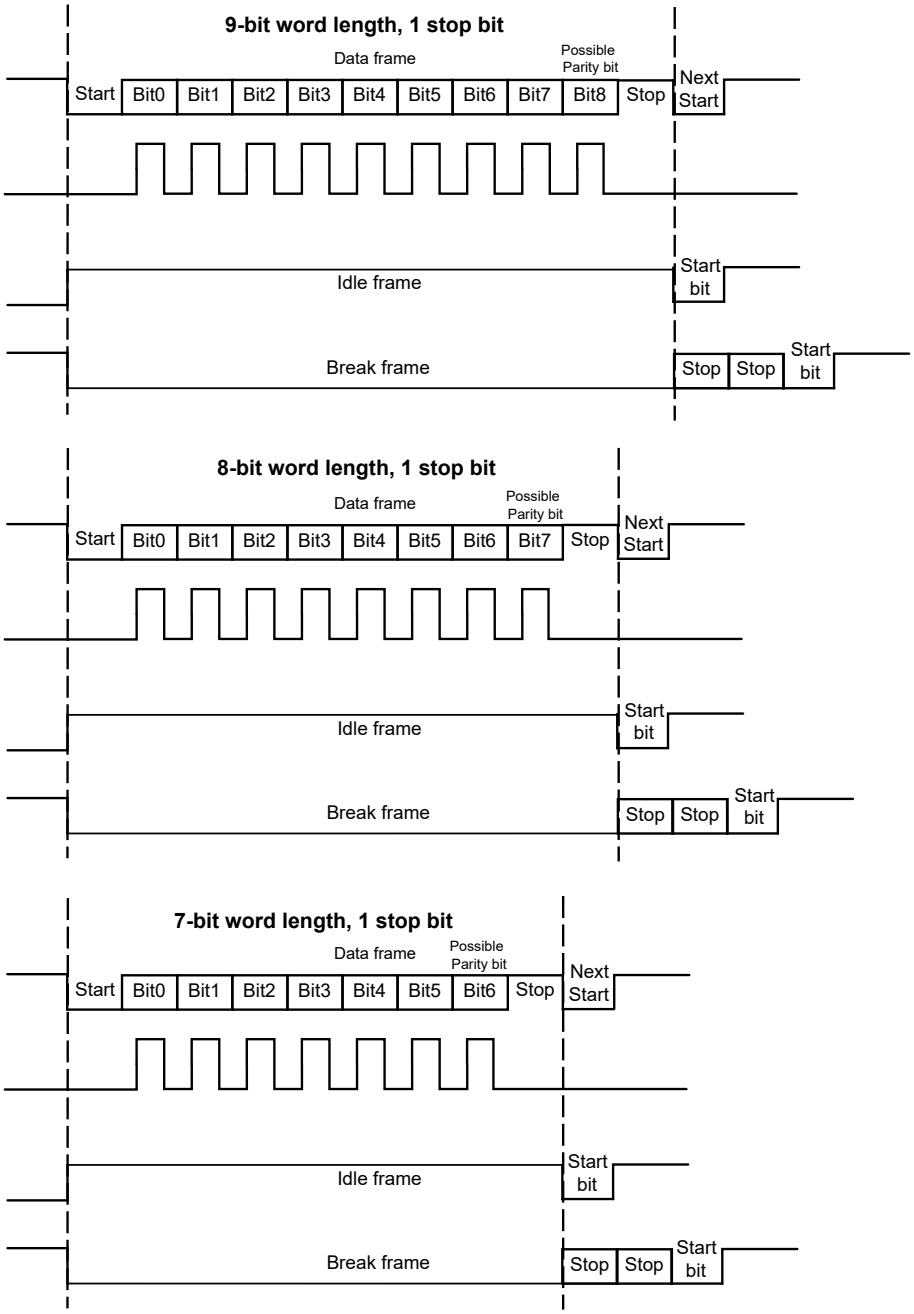


Figure 25-2 Word Length Programming

25.3.2.2 Clock Source and Oversampling Method

The choice of the clock source is done through the clock control system (refer to Reset and Clock Control (RCC)). The clock source must be selected through the UE bit before enabling the USART.

The clock source frequency is usart_ker_ck.

When the dual clock domain features are supported, the usart_ker_ck clock source can be configurable in the RCC. Otherwise, the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor defined in the USART_PRESC register.

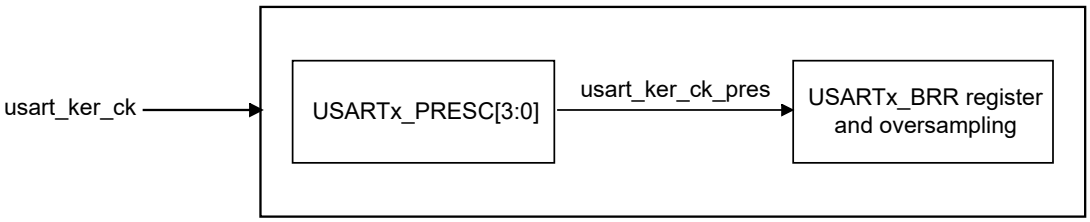


Figure 25-3 Usart_ker_ck Clock Divider Block Diagram

The communication speed range (especially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (refer to Figure 25-4 and Figure 25-5).

Depending on your application:

- Select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8).
- Select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16.

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on your application:

  – Select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected because this indicates that a glitch occurred during the sampling.
  – Select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (refer to Tolerance of the USART Receiver to Clock Deviation). In this case, the NE bit is never set.

When a frame detects noise:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in the case of single-byte communication. However, this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled), which itself generates an interrupt. In the case of multibuffer communication, an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting the NECF bit in the USART_ICR register.

NOTE: Noise error is not supported in SPI mode.

Oversampling by 8 is not available in the LIN mode. In this mode, the OVER8 bit is forced to ‘0 ’ by hardware.

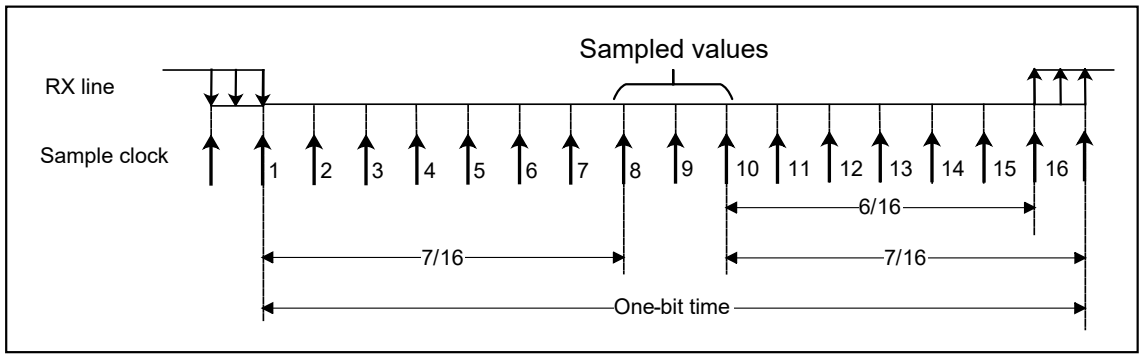


Figure 25-4 Data Sampling when Oversampling by 16

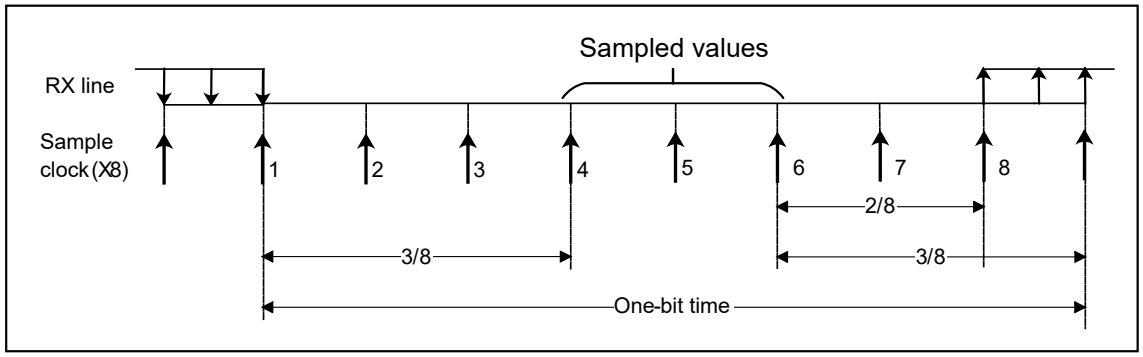


Figure 25-5 Data Sampling when Oversampling by 8

Sampled Value	NE Status	Received Bit Value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

25.3.2.3 Configurable Stop Bits

The number of stop bits with every character can be programmed in USART_CR2, bits 13,12. it can be 0.5, 1, 1.5 or 2 in normal mode.

- 0.5 stop bit
- 1 stop bit
- 1.5 stop bits
- 2 stop bits

The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

An idle frame transmission includes the stop bits.

A break transmission features 10 low bits (when M[2:0] = 000) or 11 low bits (when M[2:0] = 001) or 9 low bits (when M[2:0] = 010) followed by 2 stop bits (see Figure 25-6). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

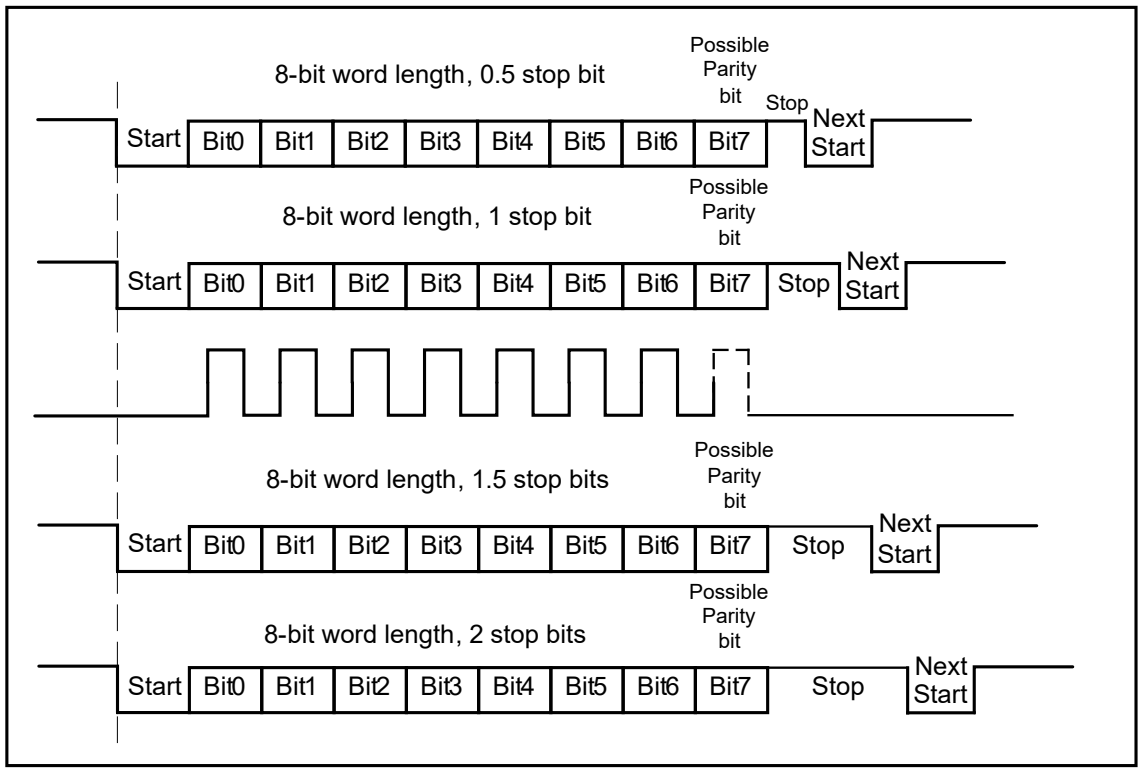


Figure 25-6 Configurable Stop Bits

25.3.2.4 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 bits, the possible USART frame formats are as listed in Table 25-1.

Table 25-1 USART Frame Formats

M Bits	PCE Bit	USART Frame
000	0	| SB(1) | 8-bit data | STB(2) |
000	1	| SB | 7-bit data | PB(3) | STB |
001	0	| SB | 9-bit data | STB |
001	1	| SB | 8-bit data | PB | STB |
010	0	| SB | 7-bit data | STB |
010	1	| SB | 6-bit data | PB | STB |
100	0	| SB | 6-bit data | STB |
100	1	| SB | 5-bit data | PB | STB |
101	0	| SB | 5-bit data | STB |
101	1	| SB | 4-bit data | PB | STB |

(1) SB: Start bit.

(2) STB: Stop bit.

(3) PB: Parity bit. In the data register, the PB always takes the MSB position (8th or 7th, depending on the M bit value).

- **Even parity**

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 4, 5, 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits are set, the parity bit is equal to 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 4, 5, 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

- **Parity checking in reception**

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

- **Parity generation in transmission**

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).

25.3.2.5 Baud Rate Generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

<table>
<tr><th>Equation</th><th>Example</th></tr>
<tr><td colspan="2">Baud Rate for Standard USART (SPI Mode Included) (OVER8 = ‘0’ or ‘1’)</td></tr>
<tr><td rowspan="2">In case of oversampling by 16, the baud rate is given by the following formula:
$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ckpres}}{\text{USARTDIV}}$
In case of oversampling by 8, the baud rate is given by the following formula:
$\text{Tx/Rx baud} = \frac{2 * \text{usart\_ker\_ckpres}}{\text{USARTDIV}}$</td><td>Example 1
To obtain 9600 baud with usart_ker_ck_pres = 48 MHz:
• In case of oversampling by 16:
USARTDIV = 48 000 000/9600
BRR = USARTDIV = 0d5000 = 0x1388
• In case of oversampling by 8:
USARTDIV = 2 * 48 000 000/9600
USARTDIV = 10 000 (0d10000 = 0x2710)
BRR[3:0] = 0x0 >> 1 = 0x0
BRR = 0x2710</td></tr>
<tr><td>Example 2
To obtain 921.6 Kbaud with usart_ker_ck_pres = 120 MHz:
• In case of oversampling by 16:
USARTDIV = 120 000 000/921 600
BRR = USARTDIV = 0d130 = 0x82
• In case of oversampling by 8:
USARTDIV = 2 * 120 000 000/921 600
USARTDIV = 260 (0d260 = 0x104)
BRR[3:0] = USARTDIV[3:0] >> 1 = 0x4 >> 1 = 0x2
BRR = 0x102</td></tr>
<tr><td colspan="2">Baud Rate in LIN Modes (OVER8 = 0)</td></tr>
<tr><td>The baud rate is given by the following formula:
$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ckpres}}{\text{USARTDIV}}$</td><td></td></tr>
</table>

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

NOTE: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence, the baud rate register value should not be changed during communication.
In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

25.3.3 FIFOs and Thresholds

The USART can operate in FIFO mode. The FIFO mode is supported only in UART and SPI modes.

The TXFIFO is 9-bit wide and the RXFIFO default width is 12 bits. The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART_CR3 register.

25.3.4 Receive Exceptions

25.3.4.1 Break Character

When a break character is received, the USART handles it as a framing error.

25.3.4.2 Idle Character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

25.3.4.3 Overrun Error

- FIFO mode disabled

  An overrun error occurs if a character is received and 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 reception.

  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 is not lost. The previous data is available by reading the USART_RDR register.
  – After the RXFF flag was set, the shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set. The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

NOTE: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities:

- If RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,
- If RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.

25.3.4.4 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_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing '1' to the FECF in the USART_ICR register.

NOTE: Framing error is not supported in SPI mode.

25.3.5 Tolerance of the USART Receiver to Clock Deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 25-2 and Table 25-3, depending on the following settings:

- 5~9 bits character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 25-2 Tolerance of the USART Receiver when BRR[3:0] = 0000

M Bits	OVER8 Bit = 0		OVER8 Bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
000	3.75%	4.375%	2.50%	3.75%
001	3.41%	3.97%	2.27%	3.41%
010	4.16%	4.86%	2.77%	4.16%
100	5.36%	6.25%	3.57%	5.36%
101	4.69%	5.47%	3.13%	4.69%

Table 25-3 Tolerance of the USART Receiver when BRR[3:0] Is Different from 0000

M Bits	OVER8 Bit = 0		OVER8 Bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
000	3.33%	3.81%	2%	3%
001	3.03%	3.46%	1.82%	2.73%
010	3.7%	4.23%	2.22%	3.33%
100	4.76%	5.44%	2.86%	4.29%
101	4.17%	4.76%	2.5%	3.75%

NOTE: The data specified in Table 25-2 and Table 25-3 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 000 (11-bit times when M = 001, 9-bit times when M = 010, 8-bit times when M = 101, 7-bit times when M = 100).

25.3.6 Auto Baud Rate Detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system uses a relatively low-accuracy clock source, and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

The modes are as follows:

Table 25-4 Auto Baud Rate Mode

Auto Baud Rate Mode	Characteristic	Descriptions
Mode 0	Any character starting with a bit at '1'	In this case, the USART measures the duration of the start bit (falling edge to rising edge).
Mode 1	Any character starting with a 10xx bit pattern	The USART measures the duration of the start and the 1st data bit.
Mode 2	A 0x7F character frame	The baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6.
Mode 3	A 0x55 character frame	The baud rate is updated first at the end of the start bit (BRs), then at the end of bit 0, and finally at the end of bit 6.

The USART_BRR register must be initialized by writing a non-zero baud rate value before the auto baud rate detection is activated.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a '0').

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

NOTE: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

25.3.7 Multiprocessor Communication

It is possible to perform USART multiprocessor communications (for instance, one of the USARTs can be the master and the others to be slaves). The master's TX output is connected to the RX inputs of the slaves. The slaves with their respective TX outputs 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 actively receives the full message contents, thus reducing redundant USART service overhead for all non-addressed receivers.

The non-addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the USART_CR1 register.

NOTE: If FIFO management is enabled and MME is already set, the MME bit should not be cleared and set again quickly (within two usart_ker_ck cycles). Otherwise, Mute mode might remain active.

When the Mute mode is enabled:

- None of the reception status bits can be set;
- All the receive interrupts are inhibited;
- The RWU bit in the USART_ISR register is set to '1'. RWU can be controlled automatically by hardware or by software through the MMRQ bit in the USART_RQR register, 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:

- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

25.3.7.1 Idle Line Detection (WAKE = 0)

The USART enters Mute mode when the MMRQ bit is written to '1' and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of Mute mode behavior using Idle line detection is given in Figure 25-7.

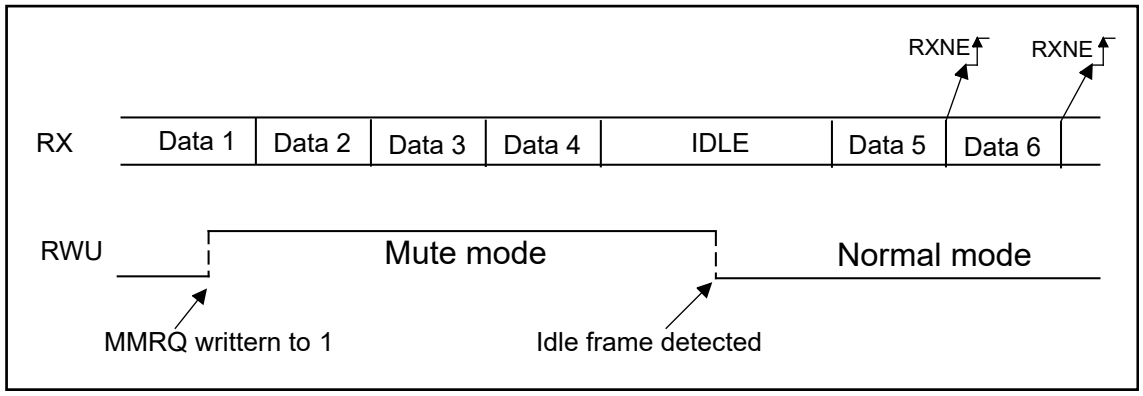


Figure 25-7 Mute Mode using Idle Line Detection

NOTE: If the MMRQ is set while the IDLE character has already elapsed, Mute mode is not entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

25.3.7.2 4-Bit/7-Bit Address Mark Detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a '1', otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7-bit or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

NOTE: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

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 or DMA request is issued when the USART enters Mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering Mute mode.

The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART 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/RXFNE bit is set for the address character since the RWU bit has been cleared.

NOTE: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in Mute mode

An example of Mute mode behavior using address mark detection is given in Figure 25-8.

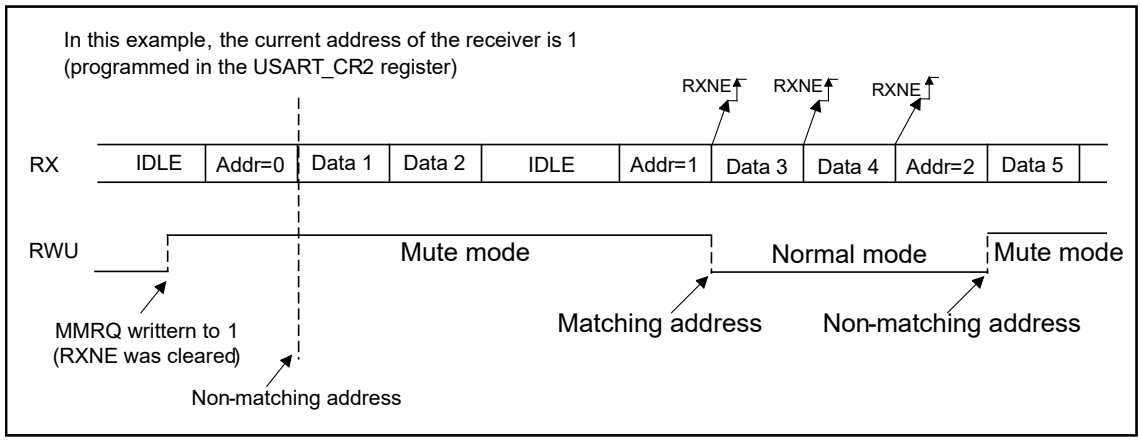


Figure 25-8 Mute Mode using Address Mark Detection

25.3.8 Modbus Communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a Half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

25.3.8.1 Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

25.3.8.2 Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

25.3.9 LIN 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:

- CLKEN in the USART_CR2 register.
- STOP[1:0], HDSEL in the USART_CR3 register.

25.3.9.1 LIN Transmission

The procedure must be the same as 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 SBKRQ bit sends 13 ‘0 bits as a break character. Then two bits of value ‘1 are sent to enable the next start detection.

25.3.9.2 LIN Reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during an 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 for 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 LBDF flag is set in USART_ISR. 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 considering 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 is 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 25-9.

Examples of break frames are given on Figure 25-10.

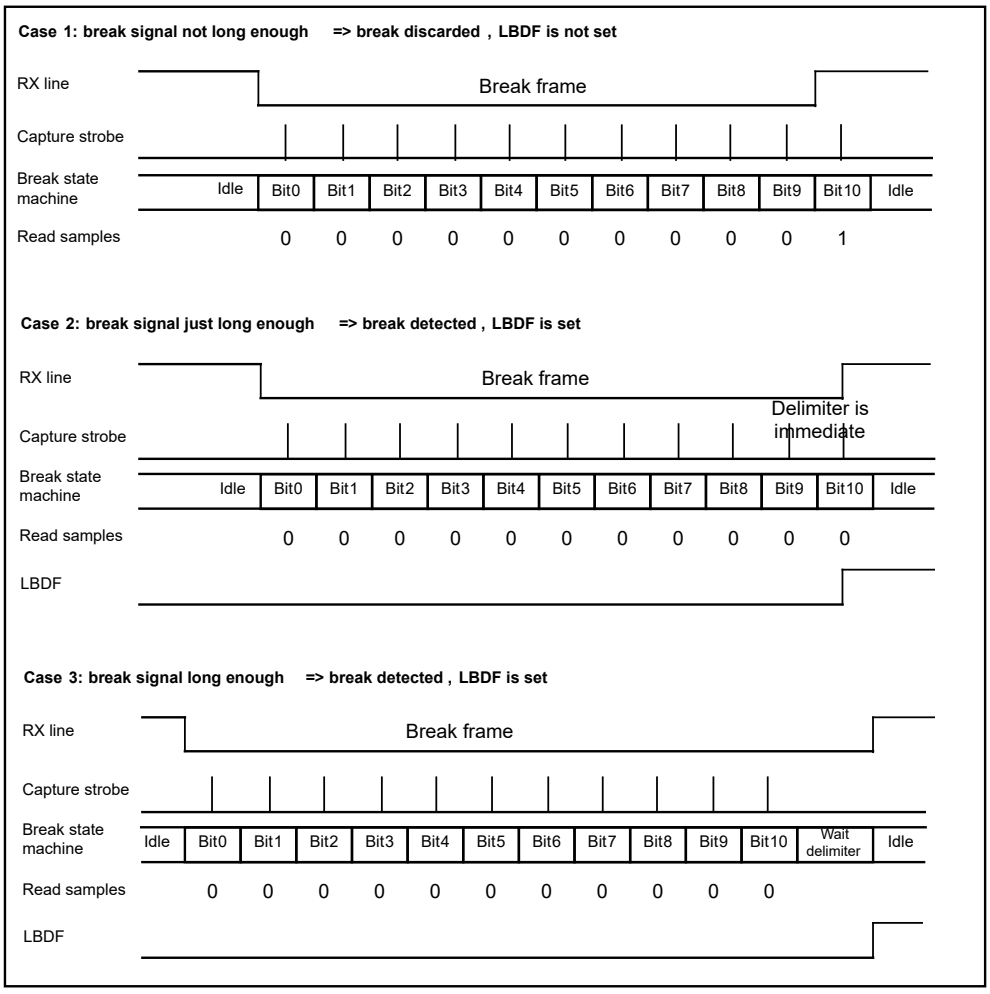


Figure 25-9 Break Detection in LIN Mode (11-Bit Break Length - LBDL Bit Is Set)

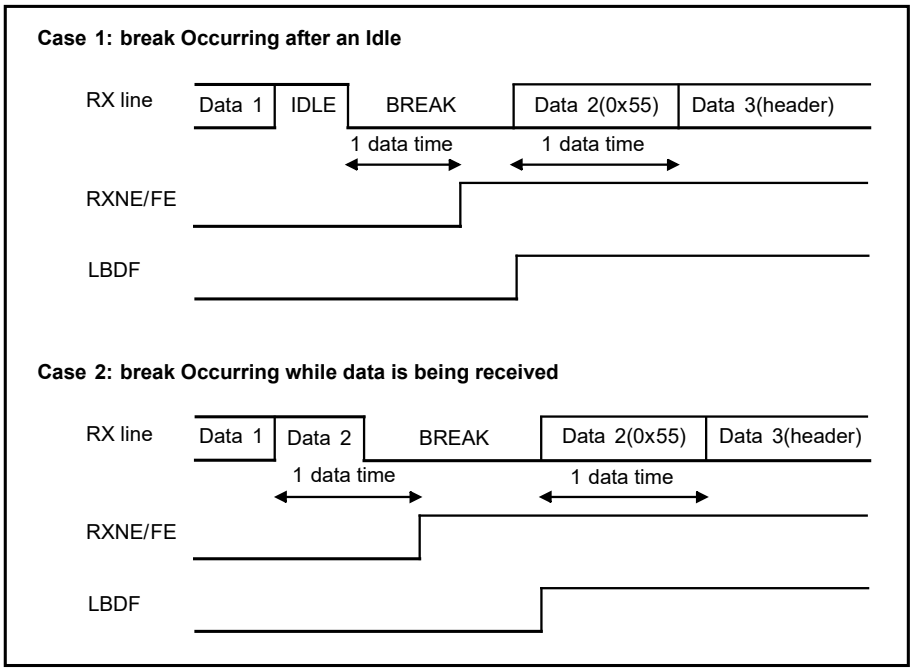


Figure 25-10 Break Detection in LIN Mode vs. Framing Error Detection

25.3.10 Synchronous Mode

The synchronous master mode is selected by programming 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.
- HDSEL bit in the USART_CR3 register.

In this mode, the USART can be used to control 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 the start and stop bits. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are or are not generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (refer to Figure 25-11, Figure 25-12, and Figure 25-13) .

During the idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

NOTE: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

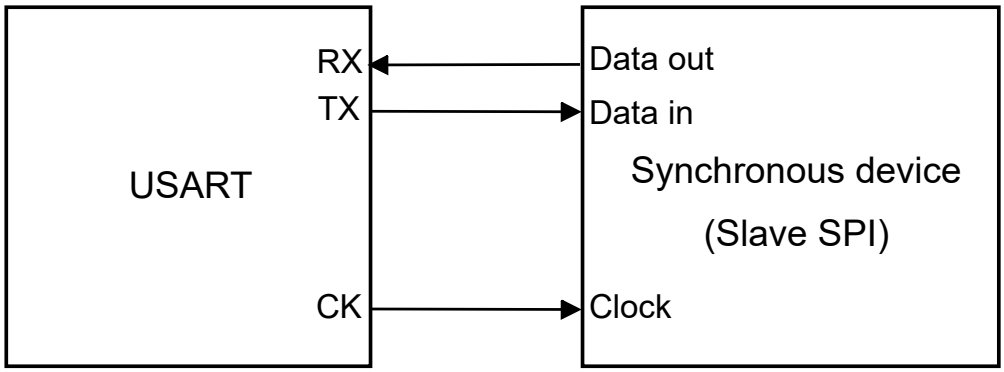


Figure 25-11 USART Example of Synchronous Master Transmission

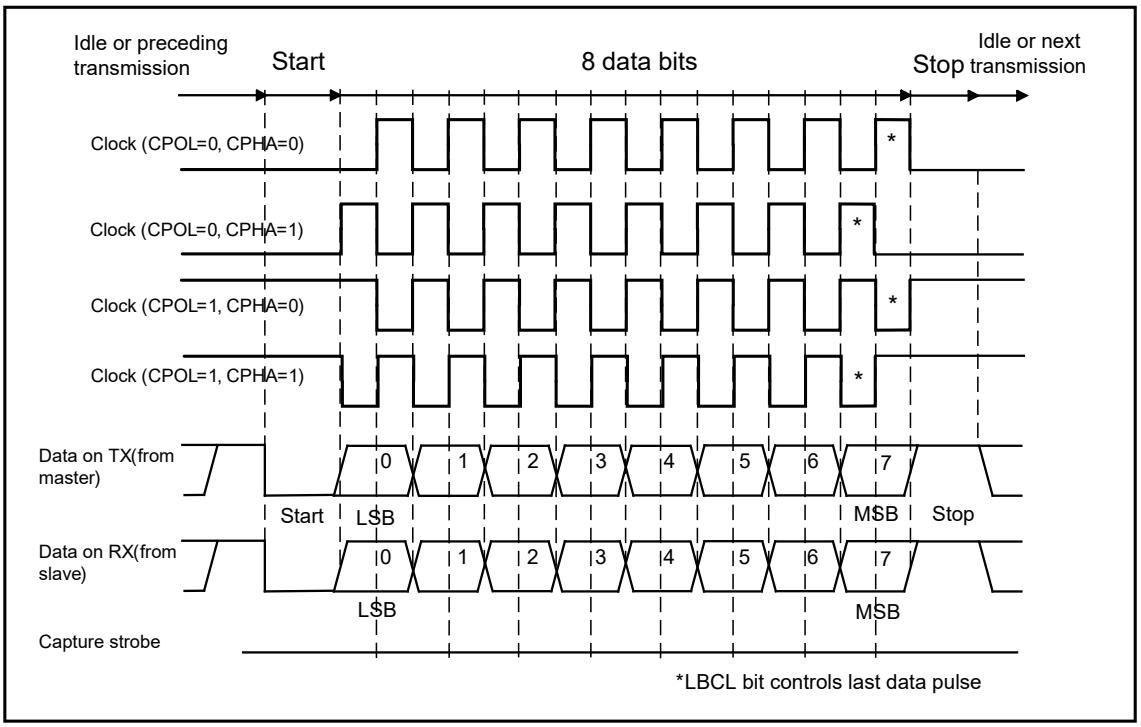


Figure 25-12 USART Data Clock Timing Diagram in Synchronous Master Mode (M Bits = 000)

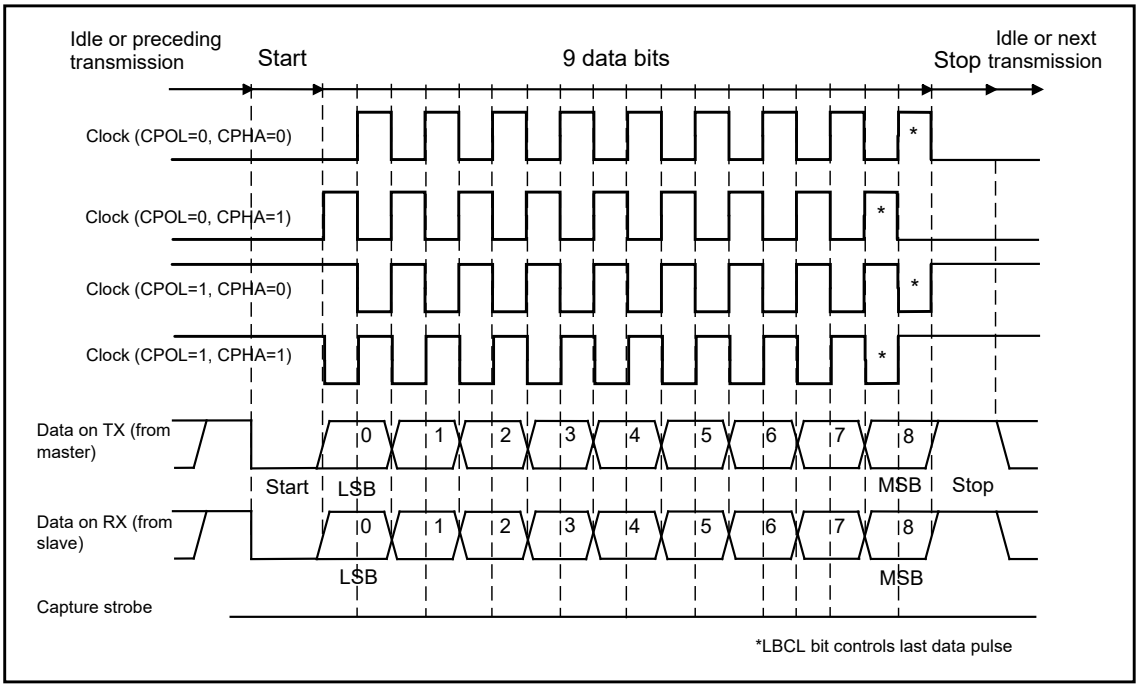


Figure 25-13 USART Data Clock Timing Diagram in Synchronous Master Mode (M bits = 001)

25.3.11 Single-Wire Half-duplex Communication

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 where the TX and RX lines are internally connected. The choice between half-duplex and full-duplex communication is determined by the control bit HDSEL in USART_CR3.

As soon as HDSEL is written to '1':

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

25.3.12 Receiver Timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register. The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = '00' or STOP = '11'
- from the end of the second stop bit if STOP = '10'.
- from the beginning of the stop bit if STOP = '01'.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

25.3.13 Continuous Communication Using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

25.3.13.1 Transmission Using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller section) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) 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 flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
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. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

NOTE: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (that is, TXFNF = 1).

25.3.13.2 Reception Using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding Direct memory access controller section) whenever a data byte is received. To map a DMA channel for USART reception, follow the steps below:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to 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.

NOTE: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

25.3.14 RS232 Hardware Flow Control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. Figure 25-14 shows how to connect 2 devices in this mode.

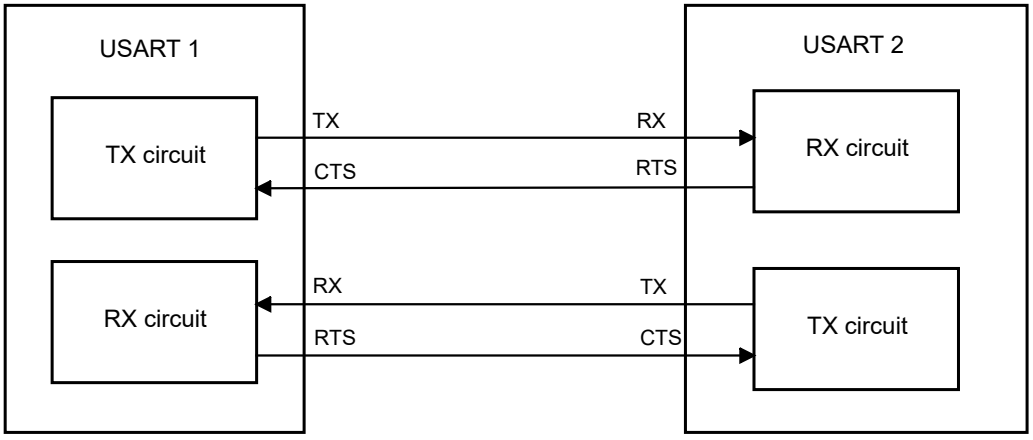


Figure 25-14 Hardware Flow Control between 2 USARTs

RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to ‘1’ in the USART_CR3 register.

The RTS is an output signal, indicating that the receiver is ready to receive data. The CTS is an input signal, and it tells the transmitter that it can send data to the receiver.

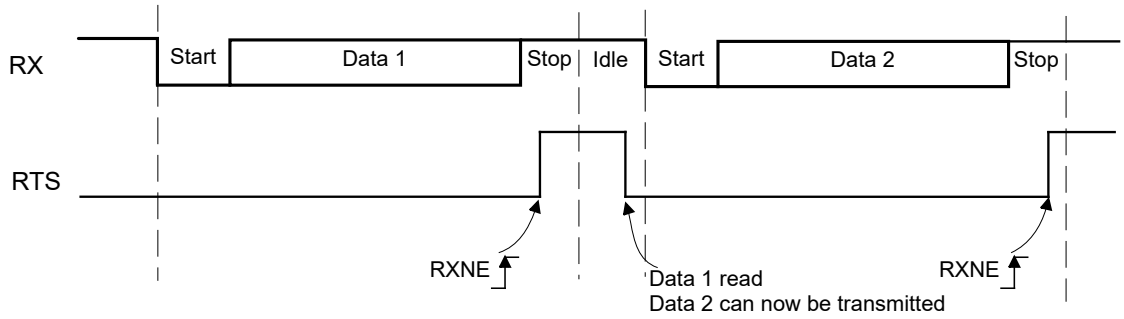


Figure 25-15 RS232 RTS Flow Control

NOTE: When FIFO mode is enabled, RTS is deasserted only when RXFIFO is full.

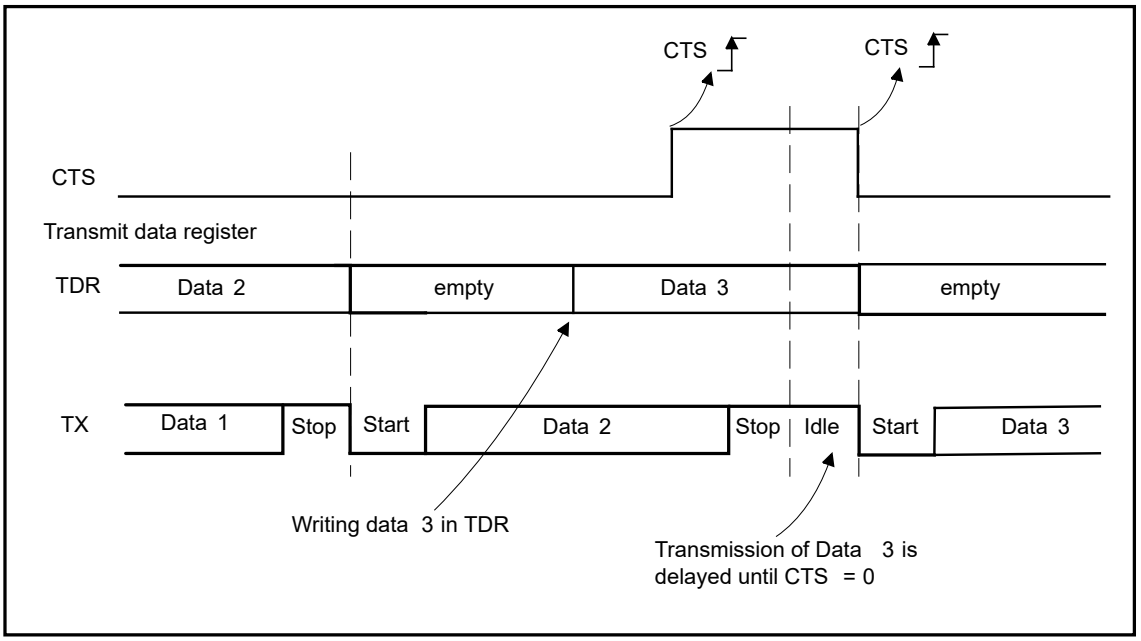


Figure 25-16 RS232 CTS Flow Control

The feature is activated in the driver by configuring the DEM bit in the USART_CR3 register. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT[4:0] bit fields in the USART_CR1 register. The deassertion time is the time between the end of the last stop bit in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT[4:0] bit fields in the USART_CR1 register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

25.3.15 Loopback Mode

In the loopback mode configuration, the transmitter output goes to the receiver input. The receiver input signal is disconnected form the UART.

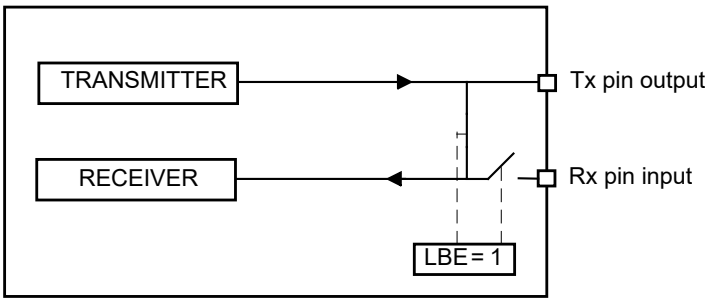


Figure 25-17 Loopback Mode Enable

Enable loopback mode by setting loopback enable bit (LBE) in the USART_CR2 register. Setting LBE disables the path from the unsynchronized receiver input signal to the receiver, and connects the transmitter output to the receiver input. In these mode, both the transmitter and receiver must be enabled.

25.4 Interrupts

Refer to Table 25-5 for a detailed description of all USART interrupt requests.

Table 25-5 USART Interrupt Requests

Interrupt Vector	Interrupt Event	Event Flag	Enable Control Bit	Interrupt Clear Method	Exit from Sleep Mode
USART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes
	Transmit FIFO not Full	TXFNF	TXFNFIE	TXFIFO full	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF	
USART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ	
	Receive FIFO Full	RXFF(1)	RXFFIE	Read RDR	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR	
	Overrun error detected	ORE	RXNEIE/ RXFNEIE	Write 1 in ORECF	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF	
	Parity error	PE	PEIE	Write 1 in PECF	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF	

Interrupt Vector	Interrupt Event	Event Flag	Enable Control Bit	Interrupt Clear Method	Exit from Sleep Mode
	Noise error in multibuffer communication	NE	EIE	Write 1 in NECF	
	Overrun error in multibuffer communication	ORE(2)		Write 1 in ORECF	
	Framing Error in multibuffer communication	FE		Write 1 in FECF	
	Character match	CMF	CMIE	Write 1 in CMCF	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF	

(1) RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).

(2) When OVRDIS = 0.

25.5 Registers

25.5.1 Register Address Map

Offset	Register Name	Register Description
0x0000	USART_CR1	USART control register 1
0x0004	USART_CR2	USART control register 2
0x0008	USART_CR3	USART control register 3
0x000c	USART_BRR	USART baud rate register
0x0014	USART_RTOR	USART receiver timeout register
0x0018	USART_RQR	USART request register
0x001c	USART_ISR	USART interrupt and status register
0x0020	USART_ICR	USART interrupt flag clear register
0x0024	USART_RDR	USART receive data register
0x0028	USART_TDR	USART transmit data register
0x002c	USART_PRESC	USART prescaler register

25.5.2 Register Field Details

25.5.2.1 USART_CR1

0x0000			USART control register 1											USART_CR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RXFFIE	TXFEIE	FIFOEN	M1	Reserved	RTOIE	DEAT					DEDT				
Type	RW	RW	RW	RW	RO	RW	RW					RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXETXFNFIE	TCIE	RXNERXFNEIE	IDLEIE	TE	RE	Reserved	UE
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-6 USART Control Register 1 Description

Field	Name	Description
31	RXFFIE	RXFIFO full interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated when RXFF = 1 in the USART_ISR register
30	TXFEIE	TXFIFO empty interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated when TXFE = 1 in the USART_ISR register
29	FIFOEN	FIFO mode enable

Field	Name	Description
		This bit is set and cleared by software. 0: FIFO mode is disabled. 1: FIFO mode is enabled. This bitfield can only be written when the USART is disabled (UE = 0). FIFO mode can be used on standard UART communication.
28	M1	Word length This bit must be used in conjunction with bit 12 in USART_CR1(M0) and bit 2 in USART_CR2 (M2) to determine the word length. It is set or cleared by software. M[2:0] = '000': 1 start bit, 8 data bits, n stop bit M[2:0] = '001': 1 start bit, 9 data bits, n stop bit M[2:0] = '010': 1 start bit, 7 data bits, n stop bit M[2:0] = '100': 1 start bit, 5 data bits, n stop bit M[2:0] = '101': 1 start bit, 6 data bits, n stop bit This bit can only be written when the USART is disabled (UE = 0). **NOTE:** In 7-bit data length mode, the LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.
27	Reserved	Reserved
26	RTOIE	Receiver timeout interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated when the RTOF bit is set in the USART_ISR register. **NOTE:** If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value.
25:21	DEAT	Driver enable assertion time

Field	Name	Description
		This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate). This bitfield can only be written when the USART is disabled (UE = 0).
20:16	DEDT	Driver enable deassertion time This 5-bit value defines the time between the end of the last stop bit in a transmitted message and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate). If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have elapsed. This bitfield can only be written when the USART is disabled (UE = 0).
15	OVER8	Oversampling mode 0: Oversampling by 16 1: Oversampling by 8 This bit can only be written when the USART is disabled (UE = 0). **NOTE:** In LIN modes, this bit must be kept clear.
14	CMIE	Character match interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated when the CMF bit is set in the USART_ISR register.
13	MME	Mute mode enable This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software. 0: Receiver in active mode permanently 1: Receiver can switch between mute mode and active mode.
12	M0	Word length

Field	Name	Description
		This bit must be used in conjunction with bit 28 in USART_CR1(M1) and bit 2 in USART_CR2(M2) to determine the word length. It is set or cleared by software. M[2:0] = '000': 1 start bit, 8 data bits, n stop bit M[2:0] = '001': 1 start bit, 9 data bits, n stop bit M[2:0] = '010': 1 start bit, 7 data bits, n stop bit M[2:0] = '100': 1 start bit, 5 data bits, n stop bit M[2:0] = '101': 1 start bit, 6 data bits, n stop bit This bit can only be written when the USART is disabled (UE = 0). **NOTE:** In 7-bit data length mode, the LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.
11	WAKE	Receiver wakeup method This bit determines the USART wakeup method from Mute mode. It is set or cleared by software. 0: Idle line 1: Address mark This bitfield can only be written when the USART is disabled (UE = 0).
10	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 = 001; 8th bit if M = 000), and the 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 transmission). 0: Parity control disabled 1: Parity control enabled This bitfield can only be written when the USART is disabled (UE = 0).
9	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.

Field	Name	Description
		0: Even parity 1: Odd parity This bitfield can only be written when the USART is disabled (UE = 0).
8	PEIE	PE interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated whenever PE = 1 in the USART_ISR register
7	TXETXFNFIE	REG MODE: TXEIE TXBUFFER empty interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated whenever TXE =1 in the USART_ISR register FIFO MODE: TXFNFIE TXFIFO not full interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated whenever TXFNF = 1 in the USART_ISR register
6	TCIE	Transmission complete interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated whenever TC = 1 in the USART_ISR register
5	RXNERXFNEIE	REG MODE: RXNEIE RXBUFFER not empty interrupt enable This bit is set and cleared by software.

Field	Name	Description
		0: Interrupt inhibited 1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register FIFO MODE: RXFNEIE RXFIFO not empty interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register
4	IDLEIE	IDLE interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register
3	TE	Transmitter enable This bit enables the transmitter. It is set and cleared by software. 0: Transmitter is disabled 1: Transmitter is enabled **NOTE:** During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word. The TE must not be immediately written to '1' to generate an idle character. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.
2	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	Reserved	Reserved
0	UE	USART enable

Field	Name	Description
		When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software. 0: USART prescaler and outputs disabled 1: USART enabled **NOTE:** The DMA requests are also reset when UE = 0, so the DMA channel must be disabled before resetting the UE bit.

25.5.2.2 USART_CR2

0x0004			USART control register 2											USART_CR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	ADDR								RTOEN	ABRMOD		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
Type	RW								RW	RW		RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SWAP	LINEN	STOP		CLKEN	CPOL	CPHA	LBCL	PM	LBDIE	LBDL	ADDM7	Reserved	M2	LBE	Reserved
Type	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RO	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-7 USART Control Register 2 Description

Field	Name	Description
31:24	ADDR	Address of the USART node These bits give the address of the USART node or a character code to be recognized. They are used to wake up the MCU with 7-bit address mark detection in multiprocessor communication during mute mode. The MSB of the character sent by the transmitter should be equal to 1. They can also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value, and the CMF flag is set on match. These bits can only be written when reception is disabled (RE = 0) or the USART is disabled (UE = 0). ADD[3:0]: These bits give the address of the USART node or a character code to be recognized. They are used for wakeup with address mark detection in multiprocessor communication during mute mode. These bits can only be written when reception is disabled (RE = 0) or the USART is disabled (UE = 0).

Field	Name	Description
23	RTOEN	Receiver timeout enable This bit is set and cleared by software. 0: Receiver timeout feature disabled 1: Receiver timeout feature enabled When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the USART_RTOR register.
22:21	ABRMOD	Auto baud rate mode These bits are set and cleared by software. 00: Measurement of the start bit is used to detect the baud rate. 01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx) 10: 0x7F frame detection 11: 0x55 frame detection This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0). **NOTE:** If DATAINV = 1 and/or MSBFIRST = 1, the patterns must be the same on the line, for example, 0xAA for MSBFIRST).
20	ABREN	Auto baud rate enable This bit is set and cleared by software. 0: Auto baud rate detection is disabled. 1: Auto baud rate detection is enabled.
19	MSBFIRST	Most significant bit first This bit is set and cleared by software. 0: Data is transmitted/received with data bit 0 first, following the start bit. 1: Data is transmitted/received with the MSB (bit 7/8) first, following the start bit. This bitfield can only be written when the USART is disabled (UE = 0).
18	DATAINV	Binary data inversion

Field	Name	Description
		This bit is set and cleared by software. 0: Logical data from the data register are sent/received in positive/direct logic. (1 = H, 0 = L) 1: Logical data from the data register are sent/received in negative/inverse logic. (1 = L, 0 = H). The parity bit is also inverted. This bitfield can only be written when the USART is disabled (UE = 0).
17	TXINV	TX pin active level inversion This bit is set and cleared by software. 0: TX pin signal works using the standard logic levels (V_{DD} = 1/idle, Gnd = 0/mark) 1: TX pin signal values are inverted. (V_{DD} = 0/mark, Gn d = 1/idle). This allows the use of an external inverter on the TX line. This bitfield can only be written when the USART is disabled (UE = 0).
16	RXINV	RX pin active level inversion This bit is set and cleared by software. 0: RX pin signal works using the standard logic levels (V_{DD} = 1/idle, Gnd = 0/mark) 1: RX pin signal values are inverted. (V_{DD} = 0/mark, Gn d = 1/idle). This allows the use of an external inverter on the RX line. This bitfield can only be written when the USART is disabled (UE = 0).
15	SWAP	Swap TX/RX pins This bit is set and cleared by software. 0: TX/RX pins are used as defined in standard pinout 1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another UART. This bitfield can only be written when the USART is disabled (UE = 0).
14	LINEN	LIN mode enable This bit is set and cleared by software. 0: LIN mode disabled 1: LIN mode enabled

Field	Name	Description
		The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN synchronous breaks. This bitfield can only be written when the USART is disabled (UE = 0).
13:12	STOP	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 bits This bitfield can only be written when the USART is disabled (UE = 0).
11	CLKEN	Clock enable This bit allows the user to enable the CK pin. 0: CK pin disabled 1: CK pin enabled
10	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 This bit can only be written when the USART is disabled (UE = 0).
9	CPHA	Clock phase This bit is used 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 This bit can only be written when the USART is disabled (UE = 0).

Field	Name	Description
8	LBCL	Last bit clock pulse This bit is used 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:** The last bit is the 5~9th data bit transmitted depending on the 5~9-bit format selected by the M* bit in the USART_CR1 + USART_CR2 register. This bit can only be written when the USART is disabled (UE = 0).
7	PM	Parity bit mask 0: Parity bit write into FIFO/buffer 1: Pairty bit mask, only data write into FIFO/buffer
6	LBDIE	LIN break detection interrupt enable Break interrupt mask (break detection using break delimiter). 0: Interrupt is inhibited 1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register
5	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 This bit can only be written when the USART is disabled (UE = 0).
4	ADDM7	7-bit Address Detection/4-bit Address Detection This bit is for selection between 4-bit address detection or 7-bit address detection. 0: 4-bit address detection 1: 7-bit address detection (in 8-bit data mode) This bit can only be written when the USART is disabled (UE = 0)

Field	Name	Description
		NOTE: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.
3	Reserved	Reserved
2	M2	Word length This bit must be used in conjunction with bit 12 in USART_CR1(M0) and bit 28 in USART_CR1(M1)to determine the word length. It is set or cleared by software. M[2:0] = '000': 1 start bit, 8 data bits, n stop bit M[2:0] = '001': 1 start bit, 9 data bits, n stop bit M[2:0] = '010': 1 start bit, 7 data bits, n stop bit M[2:0] = '100': 1 start bit, 5 data bits, n stop bit M[2:0] = '101': 1 start bit, 6 data bits, n stop bit This bit can only be written when the USART is disabled (UE = 0). **NOTE:** In 7-bit data length mode, the LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.
1	LBE	Loopback mode enable 0: Disable loopback mode 1: Enable loopback mode
0	Reserved	Reserved

25.5.2.3 USART_CR3

0x0008			USART control register 3											USART_CR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TXFTCFG			RXFTIE	RXFTCFG			Reserved	TXFTIE	Reserved						
Type	RW			RW	RW			RO	RW	RO						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	Reserved		HDSEL	Reserved		EIE
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO		RW	RO		RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-8 USART Control Register 3 Description

Field	Name	Description
31:29	TXFTCFG	TXFIFO threshold configuration 000: TXFIFO reaches 1/8 of its depth 001: TXFIFO reaches 1/4 of its depth 010: TXFIFO reaches 1/2 of its depth 011: TXFIFO reaches 3/4 of its depth 100: TXFIFO reaches 7/8 of its depth 101: TXFIFO becomes empty Remaining combinations: Reserved
28	RXFTIE	RXFIFO threshold interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited

Field	Name	Description
		1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.
27:25	RXFTCFG	Receive FIFO threshold configuration 000: Receive FIFO reaches 1/8 of its depth 001: Receive FIFO reaches 1/4 of its depth 010: Receive FIFO reaches 1/2 of its depth 011: Receive FIFO reaches 3/4 of its depth 100: Receive FIFO reaches 7/8 of its depth 101: Receive FIFO becomes full Remaining combinations: Reserved
24	Reserved	Reserved
23	TXFTIE	TXFIFO threshold interrupt enable This bit is set and cleared by software. 0: Interrupt inhibited 1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.
22:16	Reserved	Reserved
15	DEP	Driver enable polarity selection 0: DE signal is active high. 1: DE signal is active low. This bit can only be written when the USART is disabled (UE = 0). **NOTE:** If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value.
14	DEM	Driver enable mode This bit allows the user to activate the external transceiver control through the DE signal. 0: DE function is disabled. 1: DE function is enabled. The DE signal is output on the RTS pin. This bit can only be written when the USART is disabled (UE = 0).

Field	Name	Description
13	DDRE	DMA Disable on Reception Error 0: DMA is not disabled in case of a reception error. The corresponding error flag is set, but RXNE is kept at 0, preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data will be transferred. 1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE in case FIFO mode is enabled) before clearing the error flag. This bit can only be written when the USART is disabled (UE = 0). **NOTE:** The reception errors are parity, framing, or noise.
12	OVRDIS	Overrun Disable This bit is used to disable the receive overrun detection. 0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data. 1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set, the ORE flag is not set, and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO will be bypassed, and data will be written directly in the USART_RDR register. Even when FIFO management is enabled, the RXNE flag will be used. This bit can only be written when the USART is disabled (UE = 0). **NOTE:** This control bit allows checking the communication flow w/o reading the data.
11	ONEBIT	One sample bit method enable This bit allows the user to select the sample method. When the one sample bit method is selected, the noise detection flag (NE) is disabled. 0: Three sample bit method 1: One sample bit method This bit can only be written when the USART is disabled (UE = 0).
10	CTSIE	CTS interrupt enable

Field	Name	Description
		0: Interrupt is inhibited 1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register **NOTE:** If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value.
9	CTSE	CTS enable 0: CTS hardware flow control disabled 1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted. This bit can only be written when the USART is disabled (UE = 0)
8	RTSE	RTS enable 0: RTS hardware flow control disabled 1: RTS output enabled, data is only requested when there is space in the receive buffer. Data transmission is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received. This bit can only be written when the USART is disabled (UE = 0).
7	DMAT	DMA enable transmitter This bit is set/reset by software. 0: DMA mode is disabled for transmission 1: DMA mode is enabled for transmission
6	DMAR	DMA enable receiver This bit is set/reset by software. 0: DMA mode is disabled for reception 1: DMA mode is enabled for reception
5:4	Reserved	Reserved
3	HDSEL	Half-duplex selection

Field	Name	Description
		Selection of Single-wire Half-duplex mode 0: Half duplex mode is not selected 1: Half duplex mode is selected This bit can only be written when the USART is disabled (UE = 0).
2:1	Reserved	Reserved
0	EIE	Error interrupt enable Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error or overrun error noise flag error (FE = 1 or ORE = 1 or NE = 1 in the USART_ISR register). 0: Interrupt inhibited 1: Interrupt generated when FE = 1 or ORE = 1 or NE = 1 (in SPI slave mode) in the USART_ISR register.

25.5.2.4 USART_BRR

0x000c			USART baud rate register											USART_BRR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BRR															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-9 USART Baud Rate Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	BRR	USART baud rate BRR[15:4] = USARTDIV[15:4] When OVER8 = 0, BRR[3:0] = USARTDIV[3:0]. When OVER8 = 1, BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right. BRR[3] must be kept clear.

25.5.2.5 USART_RTOR

0x0014			USART receiver timeout register											USART_RTOR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved								RTO							
Type	RO								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RTO															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-10 USART Receiver Timeout Register Description

Field	Name	Description
31:24	Reserved	Reserved
23:0	RTO	Receiver timeout value This bitfield gives the Receiver timeout value in terms of number of bits during which there is no activity on the RX line. In standard mode, the RTOF flag is set if no new start bit is detected for more than the RTO value after the last received character. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character. **NOTE:** This value must only be programmed once per received character.

25.5.2.6 USART_RQR

0x0018			USART request register											USART_RQR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved											TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
Type	RO											RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-11 USART Request Register Description

Field	Name	Description
31:5	Reserved	Reserved
4	TXFRQ	Transmit data flush request When FIFO mode is disabled, writing '1' to this bit sets the TXE flag. This allows for discarding the transmitted data. When FIFO is enabled, the TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART modes. **NOTE:** In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty to ensure no data are written in the data register.
3	RXFRQ	Receive data flush request Writing 1 to this bit empties the entire receive FIFO, i.e. clears the bit RXFNE. This allows discarding the received data without reading them and avoids an overrun condition.
2	MMRQ	Mute mode request

Field	Name	Description
		Writing 1 to this bit puts the USART in Mute mode and resets the RWU flag.
1	SBKRQ	Send break request Writing 1 to this bit sets the SBKF flag and requests to send a BREAK on the line as soon as the transmit machine is available. **NOTE:** When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.
0	ABRRQ	Auto baud rate request Writing 1 to this bit resets the ABRF flag in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

25.5.2.7 USART_ISR

0x001c			USART interrupt and status register											USART_ISR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved				TXFT	RXFT	Reserved	RXFF	TXFE	Reserved	TEACK	Reserved	RWU	SBKF	CMF	BUSY
Type	RO				RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ABRF	ABRE	Reserved		RTOF	CTS	CTSIF	LBDF	TXETXFNF	TC	RXNERXFNE	IDLE	ORE	NE	FE	PE
Type	RO	RO	RO		RO	RO	WC	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0

Table 25-12 USART Interrupt and Status Register Description

Field	Name	Description
31:28	Reserved	Reserved
27	TXFT	TXFIFO threshold flag This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of the USART_CR3 register, i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE = 1 (bit 31) in the USART_CR3 register. 0: TXFIFO does not reach the programmed threshold. 1: TXFIFO reached the programmed threshold.
26	RXFT	RXFIFO threshold flag This bit is set by hardware when the threshold programmed in RXFTCFG in the USART_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE = 1 (bit 27) in the USART_CR3 register.

Field	Name	Description
		0: Receive FIFO does not reach the programmed threshold. 1: Receive FIFO reached the programmed threshold. **NOTE:** Setting the RXFTCFG threshold to '101' activates the RXFT flag when 16 data are available - 15 in the RXFIFO and 1 in the USART_RDR. As a result, the reception of the 17th data does not trigger an overrun error. However, this error occurs upon receiving the 18th data.
25	Reserved	Reserved
24	RXFF	RXFIFO full This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register. An interrupt is generated if the RXFFIE bit =1 in the USART_CR1 register. 0: RXFIFO not full 1: RXFIFO Full
23	TXFE	TXFIFO empty This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register. An interrupt is generated if the TXFEIE bit =1 (bit 30) in the USART_CR1 register. 0: TXFIFO not empty 1: TXFIFO empty
22	Reserved	Reserved
21	TEACK	Transmit enable (TE) acknowledge flag This bit is set/reset by hardware when the TE value is considered by the USART. It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, to respect the TE = 0 minimum period.
20	Reserved	Reserved
19	RWU	Receiver wakeup from mute mode

Field	Name	Description
		This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register. When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register. 0: Receiver in active mode 1: Receiver in mute mode
18	SBKF	Send break flag This bit indicates that a send break character was requested. It is set by software by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission. 0: No break character is transmitted 1: Break character will be transmitted
17	CMF	Character match flag This bit is set by hardware when a character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register. An interrupt is generated if CMIE = 1 in the USART_CR1 register. 0: No Character match detected 1: Character Match detected
16	BUSY	Busy flag This bit is set and reset by hardware. It is active when communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not). 0: USART is idle (no reception) 1: Reception ongoing
15	ABRF	Auto baud rate flag This bit is set by hardware when the automatic baud rate has been set (RXFNE will also be set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXFNE, and FE are also set in this case). It is cleared by software to request a new auto baud rate detection by writing 1 to the ABRRQ in the USART_RQR register.

Field	Name	Description
14	ABRE	Auto baud rate error This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed). It is cleared by software, by writing 1 to the ABRRQ bit in the USART_CR3 register.
13:12	Reserved	Reserved
11	RTOF	Receiver timeout This bit is set by hardware when the timeout value, programmed in the USART_RTOR register, lapsed without communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register. An interrupt is generated if RTOIE = 1 in the USART_CR2 register. 0: Timeout value not reached 1: Timeout value reached without any data reception **NOTE:** If a time equal to the value programmed in the RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), the RTOF flag is set. The counter counts even if RE = 0, but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF will be set. If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.
10	CTS	CTS flag This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin. 0: nCTS line set 1: nCTS line reset
9	CTSIF	CTS interrupt flag This bit is set by hardware when the nCTS input toggles if the CTSE bit is set. It is cleared by software by writing 1 to the CTSCF bit in the USART_ICR register. An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

Field	Name	Description
		0: No change occurred on the nCTS status line 1: A change occurred on the nCTS status line
8	LBDF	LIN break detection flag This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR. An interrupt is generated if LBDIE = 1 in the USART_CR2 register. 0: LIN Break not detected 1: LIN break detected
7	TXETXFNF	REG MODE: TXE: Transmit data register empty TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register to discard the data. An interrupt is generated if the TXEIE = 1 in the USART_CR1 register. 0: Data register full 1: Data register not full FIFO MODE: TXFNF: TXFIFO not full TXFNF is set by hardware when TXFIFO is not full, meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared, indicating that data can not be written into the USART_TDR. An interrupt is generated if the TXFNFIE bit =1 in the USART_CR1 register. 0: Transmit FIFO is full 1: Transmit FIFO is not full
6	TC	Transmission complete This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.

Field	Name	Description
		An interrupt is generated if TCIE = 1 in the USART_CR1 register. TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register. 0: Transmission is not complete 1: Transmission is complete
5	RXNERXFNE	REG MODE: RXNE: Read data register not empty RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register. An interrupt is generated if RXNEIE = 1 in the USART_CR1 register. 0: Data is not received 1: Received data is ready to be read FIFO MODE: RXFNE: RXFIFO not empty RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO. RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register. An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register. 0: Data is not received 1: Received data is ready to be read
4	IDLE	Idle line detected This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register. 0: No Idle line is detected 1: Idle line is detected
3	ORE	Overrun error

Field	Name	Description
		This bit is set by hardware when the data received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by software, writing 1 to the ORECF in the USART_ICR register. An interrupt is generated if RXFNEIE = 1 or EIE = 1 in the USART_CR1 register. 0: No overrun error 1: Overrun error is detected
2	NE	Noise detection flag This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART_ICR register. 0: No noise is detected 1: Noise is detected
1	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 software, writing 1 to the FECF bit in the USART_ICR register. An interrupt is generated if EIE = 1 in the USART_CR1 register. 0: No Framing error is detected 1: Framing error or break character is detected
0	PE	Parity error This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register. An interrupt is generated if PEIE = 1 in the USART_CR1 register. 0: No parity error 1: Parity error

25.5.2.8 USART_ICR

0x0020			USART interrupt flag clear register											USART_ICR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved														CMCF	Reserved
Type	RO														RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				RTOCF	Reserved	CTSCF	LBDCF	Reserved	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
Type	RO				RW	RO	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-13 USART Interrupt Flag Clear Register Description

Field	Name	Description
31:18	Reserved	Reserved
17	CMCF	Character match clear flag Writing 1 to this bit clears the CMF flag in the USART_ISR register.
16:12	Reserved	Reserved
11	RTOCF	Receiver timeout clear flag Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
10	Reserved	Reserved
9	CTSCF	CTS clear flag Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
8	LBDCF	LIN break detection clear flag

Field	Name	Description
		Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
7	Reserved	Reserved
6	TCCF	Transmission complete clear flag Writing 1 to this bit clears the TC flag in the USART_ISR register.
5	TXFECF	TXFIFO empty clear flag Writing 1 to this bit clears the TXFE flag in the USART_ISR register.
4	IDLECF	Idle line detected clear flag Writing 1 to this bit clears the IDLE flag in the USART_ISR register.
3	ORECF	Overrun error clear flag Writing 1 to this bit clears the ORE flag in the USART_ISR register.
2	NECF	Noise detected clear flag Writing 1 to this bit clears the NE flag in the USART_ISR register.
1	FECF	Framing error clear flag Writing 1 to this bit clears the FE flag in the USART_ISR register.
0	PECF	Parity error clear flag Writing 1 to this bit clears the PE flag in the USART_ISR register.

25.5.2.9 USART_RDR

0x0024			USART receive data register											USART_RDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							RDR								
Type	RO							RO								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-14 USART Receive Data Register Description

Field	Name	Description
31:9	Reserved	Reserved
8:0	RDR	Receive data value Contains the received data character. The RDR register provides the parallel interface between the input shift register and the internal bus.

25.5.2.10 USART_TDR

0x0028			USART transmit data register											USART_TDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							TDR								
Type	RO							RW								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-15 USART Transmit Data Register Description

Field	Name	Description
31:9	Reserved	Reserved
8:0	TDR	Transmit data value Contain the data character to be transmitted. The USART_TDR register provides the parallel interface between the internal bus and the output shift register. 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.

25.5.2.11 USART_PRESC

0x002c			USART prescaler register											USART_PRESC		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												PRESCALER			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 25-16 USART Prescaler Register Description

Field	Name	Description
31:4	Reserved	Reserved
3:0	PRESCALER	Clock prescaler The USART input clock can be divided by a prescaler factor: 0000: Input clock not divided 0001: Input clock divided by 2 0010: Input clock divided by 4 0011: Input clock divided by 6 0100: Input clock divided by 8 0101: Input clock divided by 10 0110: Input clock divided by 12 0111: Input clock divided by 16 1000: Input clock divided by 32 1001: Input clock divided by 64

Field	Name	Description
		1010: Input clock divided by 128 1011: Input clock divided by 256 Remaining combinations: Reserved

Serial Peripheral Interface (SPI)/Integrated Interchip Sound (I2S)

This chapter describes the details of the Serial Peripheral Interface (SPI)/Integrated Interchip Sound (I2S).

Topics: **Page**

26.1 Introduction

The Serial Peripheral Interface (SPI)/Integrated Interchip Sound (I2S) interface enables communication with external devices using either the SPI protocol or the I2S audio protocol, with the mode being software-selectable. By default, the SPI Motorola mode is selected after a device reset.

The SPI protocol supports half-duplex, full-duplex, and simplex synchronous serial communication with external devices. When configured as master, the interface provides the communication clock (SCK) to the external slave device and can also function in a multimaster configuration.

Like SPI, the I2S protocol is a synchronous serial communication interface, capable of operating in slave or master mode with half-duplex or full-duplex communication. It supports four distinct audio standards: the Philips I2S standard, the MSB- and LSB-justified standards, and the PCM standard.

26.2 Features

26.2.1 SPI Features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4 to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to $f_{PCLK}/2$
- Slave mode frequency up to $f_{PCLK}/2$
- 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
- SPI Motorola support
- 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 flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- Enhanced TI and NSS pulse modes support
- Support clock dummy function, 1-8 configurable clock cycle to not transfer data after chip select asserts
- Support one RX clock edge delay for data receive
- Support big and little endian for byte transfer
- Under master mode, support NSS (chip select) asserts before data transfer and NSS maintain assertion after data transfer

26.2.2 I2S Features

- Full-duplex communication
- 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, 20-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 20-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode, overrun flag in reception mode (master and slave) and frame error flag in reception and transmitter mode (slave only)
- 32-bit register for transmission and reception with one data register for both channel sides (for buffer mode, only 16-bit LSB of the register used; for FIFO mode, entire 32-bit of the register is used)
- Supported I2S protocols:
  - I2S Philips 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 for buffer mode and 32-bit wide for FIFO mode)
- Master clock can be output to drive an external audio component. Ratio is fixed at 256 × F_S (where F_S is the audio sampling frequency)
- Support Tx clock inverted edge output
- Support one Rx clock edged delay for data transfer
- Support big and little endian for data transfer
- Non-PCM mode supports to choose sending right or left channel data first

26.3 Implementation

Table 26-1 describes all the SPI instances and their features embedded in the devices.

Table 26-1 SPI/I2S Implementation

SPI Features	SPI_I2S0	SPI_I2S1	SPI_I2S2
Enhanced NSSP & TI modes	Yes	Yes	Yes
Hardware CRC calculation	Yes	Yes	Yes
I2S support	Yes	Yes	Yes
Data size configurable	From 4 to 16-bit	From 4 to 16-bit	From 4 to 16-bit
Rx/Tx FIFO size	8x32-bit	8x32-bit	8x32-bit

26.4 Functional Description

26.4.1 SPI Functional Description

26.4.1.1 Block Diagram

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main elements of SPI and their interactions are shown in Figure 26-1.

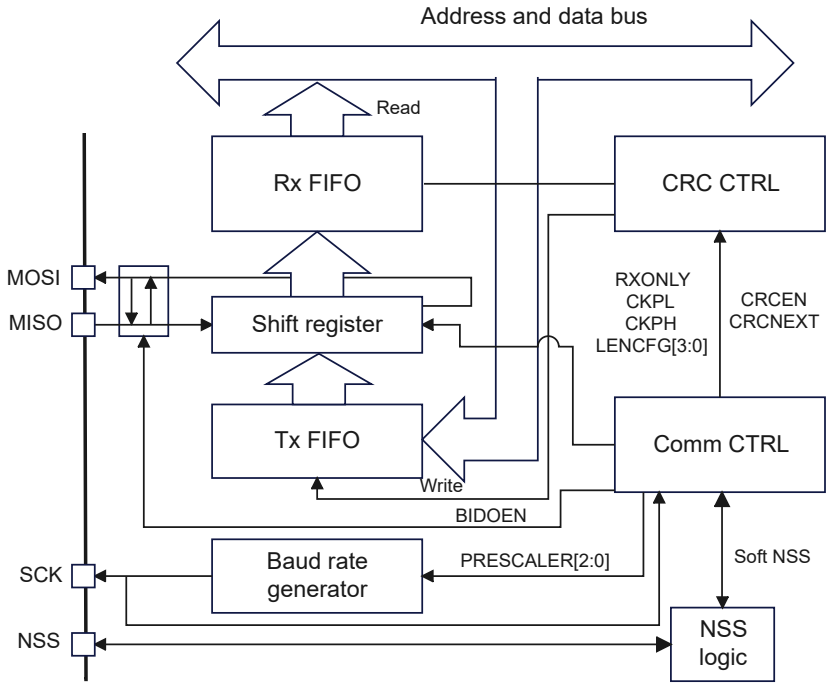


Figure 26-1 SPI Block Diagram

26.4.1.2 Signal Description

Four I/O pins are dedicated to SPI communication with external devices.

Pin Name	Direction	Description
MISO	I/O	Master In/Slave Out data In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
MOSI	I/O	Master Out/Slave In data In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
SCK	I/O	Serial Clock output pin for SPI masters and input pin for SPI slaves.
NSS	I/O	Slave select pin Depending on the SPI and NSS settings, this pin can be used to either: • Select an individual slave device for communication. • Synchronize the data frame. • Detect a conflict between multiple masters.

Pin Name	Direction	Description
		For more details, refer to Slave Select (NSS) Pin Management.

The SPI bus allows communication between one master device and one or more slave devices. The bus comprises at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

26.4.1.3 Communications between One Master and One Slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management).

Communication is always initiated by the master.

26.4.1.3.1 Full-duplex Communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

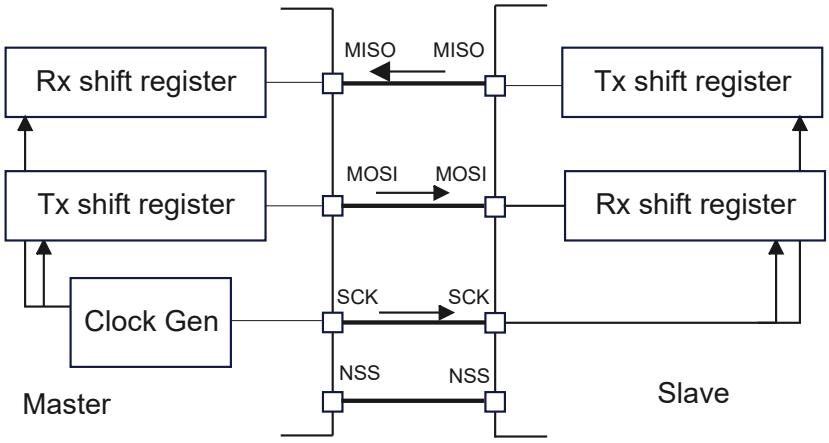


Figure 26-2 Full-duplex Single Master/Single Slave Application

26.4.1.3.2 Half-duplex Communication

The SPI can communicate in half-duplex mode by setting the BIDEN bit in the SPIx_CTL0 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected by both master and slave with the BIDOEN bit in their SPIx_CTL0 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

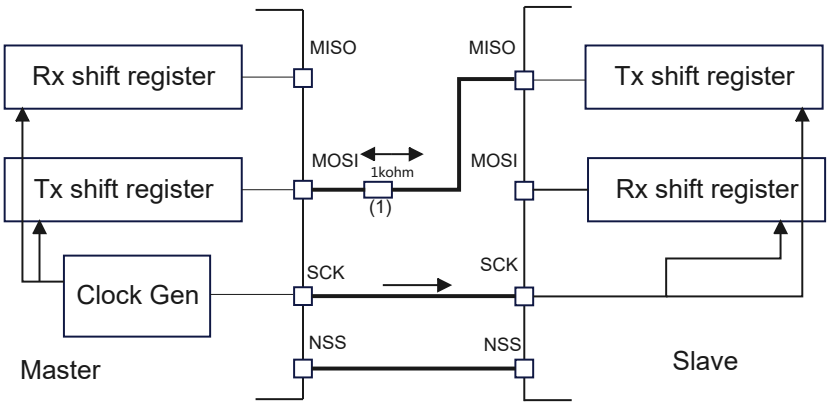


Figure 26-3 Half-duplex Single Master/Single Slave Application

(1) It is recommended to insert a serial resistor between the MISO and MOSI pins. This will help protect the outputs if the communication direction between two nodes is not changed simultaneously when operating in bidirectional mode.

26.4.1.3.3 Simplex Communication

The SPI can communicate in simplex mode by setting the SPI in transmit-only or receive-only using the RXONLY bit in the SPIx_CTL0 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair are not used for communication and can be used as standard GPIOs.

- Transmit-only mode (RXONLY = 0): The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin if the RXMASK bit is cleared. This pin can be used as a standard GPIO.
- Receive-only mode (RXONLY = 1): The application can disable the SPI output function by setting the RXONLY bit.
  - In slave configuration, the MISO output is disabled, and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active.
  - In the master configuration, the MOSI output is disabled, and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled.

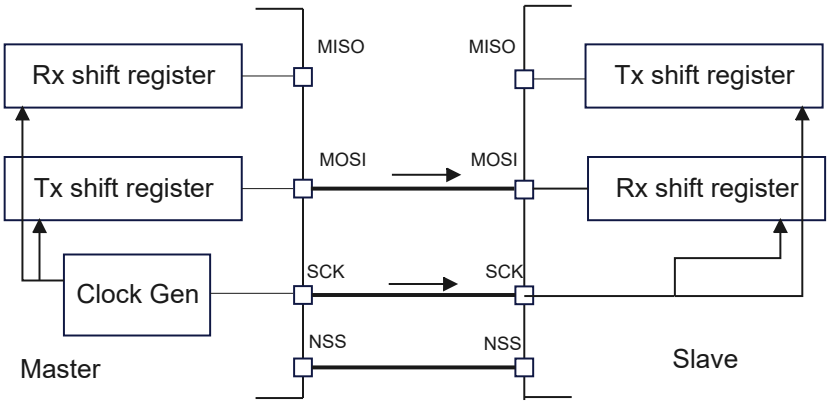


Figure 26-4 Simplex Single Master/Single Slave Application

26.4.1.4 Standard Multi-slave Communication

In a configuration with two or more independent slaves, the master uses NSSx pins to manage the chip select lines for each slave (as shown in Figure 26-5). The master configures which NSSx pin to

assert with the NSSSEL bits in SPIx_EXTCFG2. Once this is accomplished, a standard master and dedicated slave communication is established.

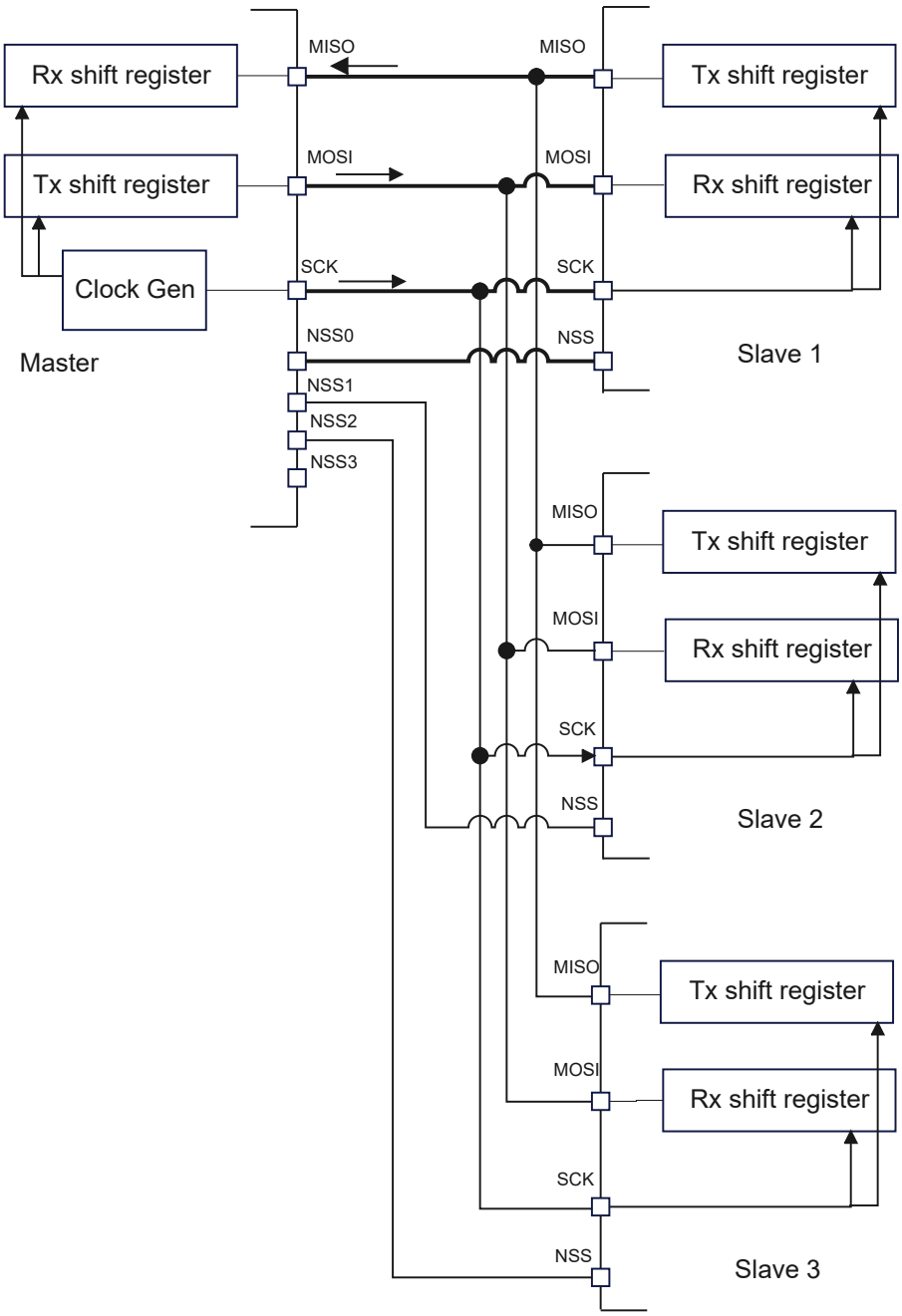


Figure 26-5 One Master and Three Independent Slaves

(1) As the MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (refer to Alternate Function I/O).

26.4.1.5 Multi-master Communication

The SPI bus is not primarily designed to support multi-master functionality. Yet, it does have an inherent feature that allows users to identify a possible conflict between two nodes attempting to gain control of the bus simultaneously. The NSS pin, configured in hardware input mode, is used for this conflict detection.

When nodes are non-active, both stay in slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via a dedicated GPIO pin. After the session is completed, the active slave select signal is released, and the node mastering the bus temporarily returns to passive slave mode, waiting for the next session to start.

If both nodes potentially raise their mastering request simultaneously, a bus conflict event appears (refer to Mode Fault (MODEFAULT) event). Users can apply a simple arbitration process to overcome this conflict.

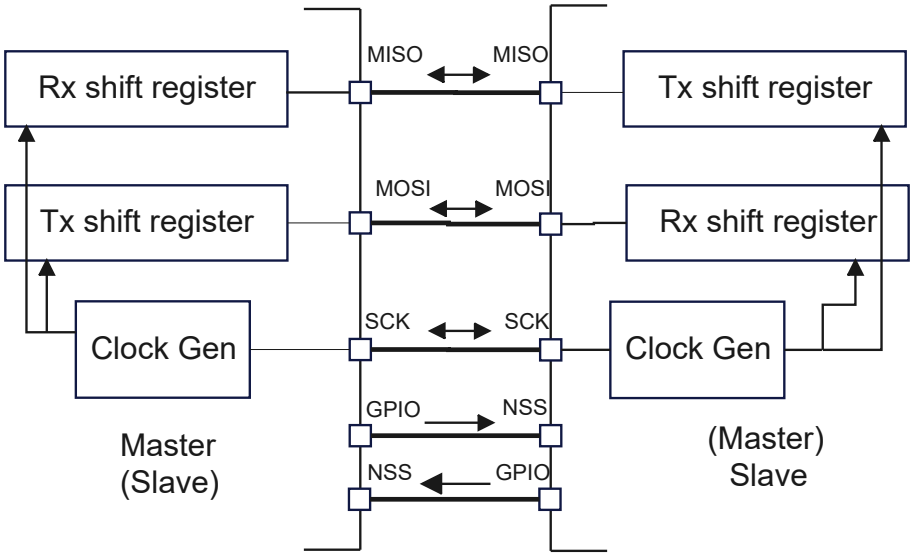


Figure 26-6 Multi-master Application

26.4.1.6 Slave Select (NSS) Pin Management

In slave mode, the NSSx pins work as a standard “chip select” input and let the slave communicate with the master. In master mode, NSSx pins can be used either as output or input. As an input, it can prevent multimaster bus collision, and as an output, it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SWNSSEN bit in the SPIx_CTL0 register:

- Software NSS management (SWNSSEN = 1): in this case, there are two possible configurations. The configuration used depends on the NSS output configuration NSSOESEL bit in register SPIx_CTL1).
  - NSS output enable (SWNSSEN = 1, NSSOESEL = 1): this configuration is only used when the MCU is set as master. The NSS pin is driven by the SWNSS bit value in register SPIx_CTL0.
  - NSS output disable (SWNSSEN = 1, NSSOESEL = 0): this configuration is only used when the MCU is set as a slave. In this configuration, slave select information is driven internally by the SWNSS bit value in register SPIx_CTL0. The external NSSx pin is free for other application uses.
- Hardware NSS management (SWNSSEN = 0): in this case, there are two possible configurations. The configuration used depends on the NSS output configuration NSSOESEL bit in register SPIx_CTL1).
  - NSS output enable (SWNSSEN = 0, NSSOESEL = 1): this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (EN = 1) and is kept low until the SPI is disabled (EN = 0). A pulse can be generated between continuous communications if NSS

pulse mode is activated (NSSMODE = 1). The SPI cannot work in multimaster configuration with this NSS setting.

– NSS output disable (SWNSSEN = 0, NSSOESEL = 0): if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters the master mode fault state, and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while the NSS line is at a low level.

Configuration		Master	Slave
SWNSSEN = 1	NSSOESEL = 1	NSS output depend on SWNSS	NSS output depend on GPIO configuration
	NSSOESEL = 0	Conflict	NSS input depend on SWNSS
SWNSSEN = 0	NSSOESEL = 1	NSS output depend on hardware NSS	NSS output depend on GPIO configuration
	NSSOESEL = 0	NSS input depend on pad, maybe conflict	NSS input depend on pad

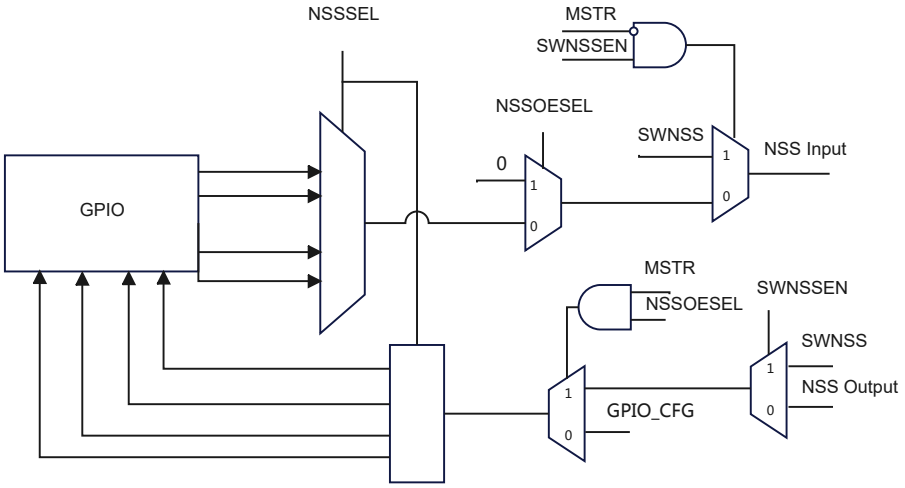


Figure 26-7 Hardware/Software Slave Select Management

26.4.1.7 Communication Format

In SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, clock polarity, and data frame format. For successful communication, the master and slave devices must adhere to the same communication format.

26.4.1.7.1 Clock Phase and Polarity Controls

Four possible timing relationships may be chosen by software, using the CKPL and CKPH bits in the SPIx_CTL0 register. The CKPL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CKPL is reset, the SCK pin has a low-level idle state. If CKPL is set, the SCK pin has a high-level idle state.

If the CKPH bit is set, the second edge on the SCK pin captures the first data bit transacted. Data are latched on each occurrence of this clock transition type. If the CKPH bit is reset, the first edge on the SCK pin captures the first data bit transacted. Data are latched on each occurrence of this clock transition type.

The combination of CKPL (clock polarity) and CKPH (clock phase) bits selects the data capture clock edge.

Figure 26-8 and Figure 26-9 show an SPI full-duplex transfer with four different combinations of CKPL and CKPH.

NOTE: Prior to changing the CKPL/CKPH bits, the SPI must be disabled by resetting the EN bit.

The idle state of SCK must correspond to the polarity selected in the SPIx_CTL0 register (by pulling up SCK if CKPL = 1 or pulling down SCK if CKPL = 0).

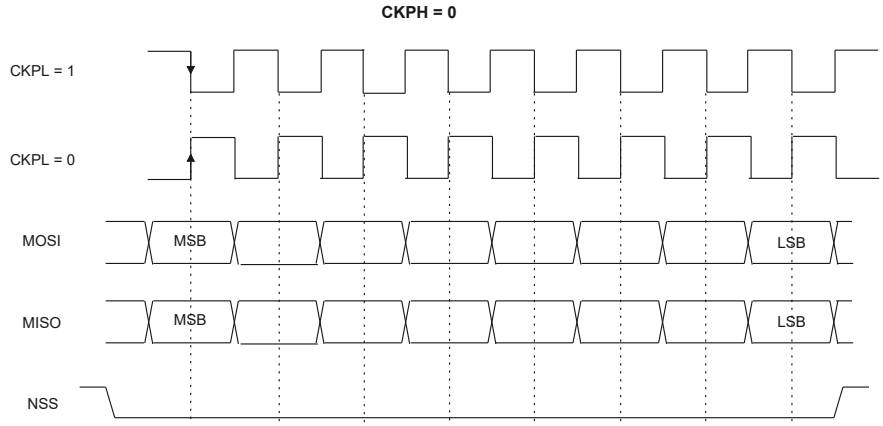


Figure 26-8 Data Clock Timing Diagram – CKPH = 0

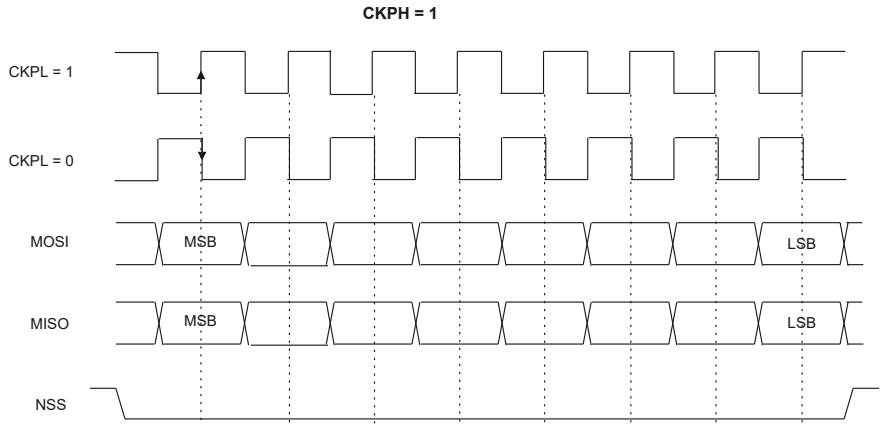


Figure 26-9 Data Clock Timing Diagram – CKPH = 1

26.4.1.7.2 Data Frame Format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBF bit. The data frame size is chosen either by FRMFORMAT bit to be 8-bit or 16-bit data format, or by using the LENCFG[3:0] bits when LENCFGEN = 1, in this case, it can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception.

When the SPIx_DATA register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte) or a half-word. During communication, only bits within the data frame are clocked and transferred.

26.4.1.8 SPI Configuration

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated sections. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: configure GPIO for NSSx, MOSI, MISO and SCK pins.
2. Write to the NSSSEL bits in SPIx_EXTCFG2 to choose which NSSx pin to assert for master mode.
3. Write to the SPIx_CTL0 register:
   a. Configure the serial clock baud rate using the PRESCALER[2:0] bits.
   b. Configure the CKPL and CKPH bits combination to define one of the four relationships between the data transfer and the serial clock.
   c. Select simplex or half-duplex mode by configuring RXONLY or BIDEN and BIDOEN.
   d. Configure the LSBF bit to define the frame format.
   e. Configure the CRCEN bit if CRC is needed.
   f. Configure SWNSSEN and SWNSS if software NSS is needed.
   g. Configure the MSTR bit.
4. Write to SPIx_CTL1 register:
   a. Enable LENCFGEN to bypass FRMFORMAT and configure the LENCFG[3:0] bits to select the data length for the transfer.
   b. Configure NSSOESEL to enable NSS output for SPI master
   c. Set the TIMODE bit if the TI protocol is required (keep NSSMODE bit cleared in TI mode)
   d. Set the NSSMODE bit if the NSS pulse mode between two data units is required (keep CKPH and TIMODE bits cleared in NSS pulse mode).
5. Write to SPIx_CRCPOLY register for CRC polynomial, write to SPIx_CRCCFG for CRC initial value and write to the CRCXOROUT bit in SPIx_EXTCFG1 if needed to XOR CRC output with either 0x0 or 0xFFFF.
6. Write DMA registers dedicated to SPI Tx and Rx if DMA transfers are used.
7. Optionally, if required, write to SPIx_EXTCFG2 and SPIx_EXTCFG3 to enable and select SCK delay, data delay or NSS delay for SPI transfers.

26.4.1.9 Data Transmission and Reception Procedures

26.4.1.9.1 Rx and Tx Buffers (Buffer Mode)

When the FIFOBUFSEL bit in SPI_EXTCFG0 = 0, FIFO mode is disabled, data transmission uses 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 SPIx_DATA register returns the Rx buffered value whereas a write access to the SPIx_DATA stores the written data into the Tx buffer.

Handling data transmission and reception

The TXEFLAG bit in SPIx_STAT (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 TXFEIE bit in the SPIx_CTL1 register is set. Clearing the TXEFLAG bit is performed by writing to the SPIx_DATA register.

The RXNEFLAG bit in SPIx_STAT (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 SPIx_DATA register. An interrupt can be generated if the RXFNEIE bit in the SPI_CTL1 register is set. Clearing the RXNEFLAG bit is performed by reading the SPIx_DATA register.

For some configurations, the BUSY flag can be used during the last data transfer to wait until the completion of the transfer.

26.4.1.9.2 RXFIFO and TXFIFO (FIFO Mode)

When the FIFOBUFSEL bit in SPI_EXTCFG0 = 1, FIFO mode is enabled, all SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO.

A read access to the SPIx_DATA register returns the oldest value stored in RXFIFO that has not been read yet.

A write access to the SPIx_DATA stores the written data in the TXFIFO at the end of a send queue. RXFAFLVL and TXFAELVL in SPIx_FIFOCFG register represents RXFIFO and TXFIFO threshold configuration. A read access to the SPIx_DATA register must be managed by the RXNEFLAG event. This event is triggered when data is stored in RXFIFO and the threshold (defined by RXFAFLVL bit) is reached. When RXFEMPTYFLAG is set, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXFAEMPTYFLAG event. This event is triggered when the TXFIFO level is less than threshold defined by TXFAELVL bit. Both TXFAEMPTYFLAG and RXNEFLAG events can be polled or handled by interrupts.

If the next data is received when the RXFIFO is full, an overrun event occurs (refer to the description of RXFOVERFLOW flag at SPI Status Flags). An overrun event can be polled or handled by an interrupt.

The BUSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BUSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

26.4.1.9.3 Procedure for Enabling SPI

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 must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full-duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY = 1 or BIDIMODE = 1 & BIDIOE = 0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated section.

26.4.1.9.4 Procedure for Disabling SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped.

Master in full-duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Before the SPI is disabled in these modes, the user must follow standard disable procedure.

When the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by clearing EN bit. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts. Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (refer to the SPIxRST bits in the RCC_APBxRSTR registers).

Standard disable procedure is based on pulling BUSY status together with TXEFLAG to check if a transmission session is fully completed.

The correct disable procedure is (except when receive only mode is used):

1. Wait until TXEFLAG = 1 (no more data to transmit).
2. Wait until BUSY = 0 (the last data frame is processed).
3. Disable the SPI (EN = 0).
4. Read data until RXFEMPTYFLAG = 1 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (EN = 0) in the specific time window while the last data frame is ongoing.
2. Wait until BUSY = 0 (the last data frame is processed).
3. Read data until RXFEMPTYFLAG = 1 (read all the received data).

26.4.1.9.5 Communication Using DMA

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 DMATXEN or DMARXEN bits in the SPI_CTL1 register are enabled. Separate requests must be issued to the Tx and Rx buffers.

In transmission, a DMA request is issued each time TXEFLAG is set to 1 for buffer mode or TXFAEMPTYFLAG is set to 1 for FIFO mode. The DMA then writes to the SPIx_DATA register (this clears the TXEFLAG or TXFAEMPTYFLAG bit based on buffer mode or FIFO mode selected).

In reception, a DMA request is issued each time RXNEFLAG is set to 1. The DMA then reads the SPIx_DATA register (this clears the RXNEFLAG bit).

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the RXFOVERFLOW 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 CHxFULLTRANSFERINT is set in the DMA_INTSTSx register), the BUSY 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 TXEFLAG = 1 and then until BUSY = 0.

26.4.1.9.6 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 not automatic and needs software intervention to set CRCNEXT bit. After the CRC reception, the CRC must be read in the SPIx_DATA register to clear the RXNEFLAG bit.

At the end of data and CRC transfers, the CRCERR flag in SPIx_STAT is set if corruption occurs during the transfer.

26.4.1.10 NSS Pulse Mode

This mode is activated by the NSSMODE bit in the SPIx_CTL1 register and it takes effect only if the SPI interface is configured as Motorola SPI master (TIMODE = 0) with clock polarity and phase setting as CKPL = 1 and CKPH = 0. When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data.

Figure 26-10 illustrates NSS pin management when NSSP pulse mode is enabled.

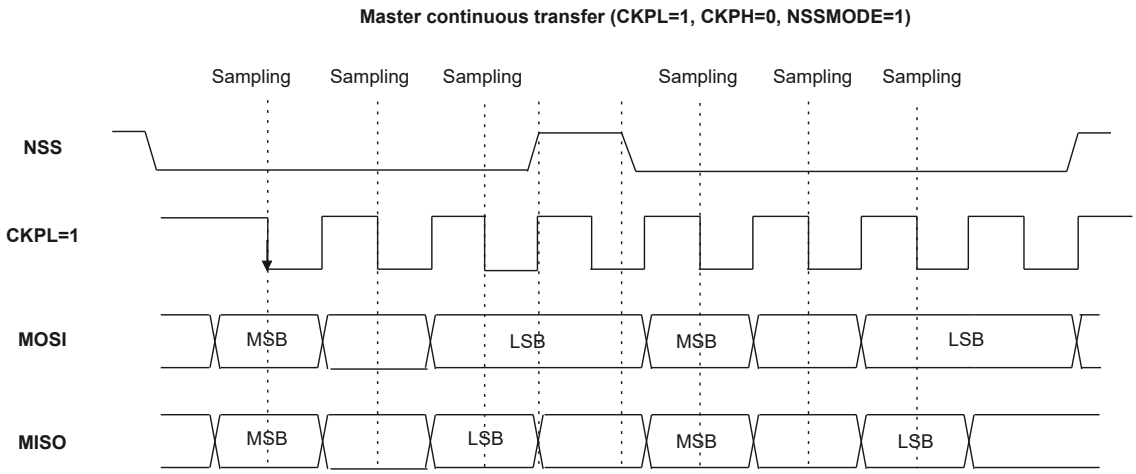


Figure 26-10 NSSP Pulse Generation in Motorola SPI Master Mode

NOTE: The similar behavior is encountered when CKPL = 0. In this case, the sampling edge is the rising edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

26.4.1.11 TI Mode

TI protocol in master mode

The SPI interface is compatible with the TI protocol. The TIMODE bit of the SPIx_CTL1 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase needs to be configured as CKPL=0 and CKPH=1 for TI mode communication.

If the slave detects a misplaced NSS pulse during a data frame transaction the FRMERR flag is set.

Figure 26-11 shows the SPI communication waveforms when TI mode is selected.

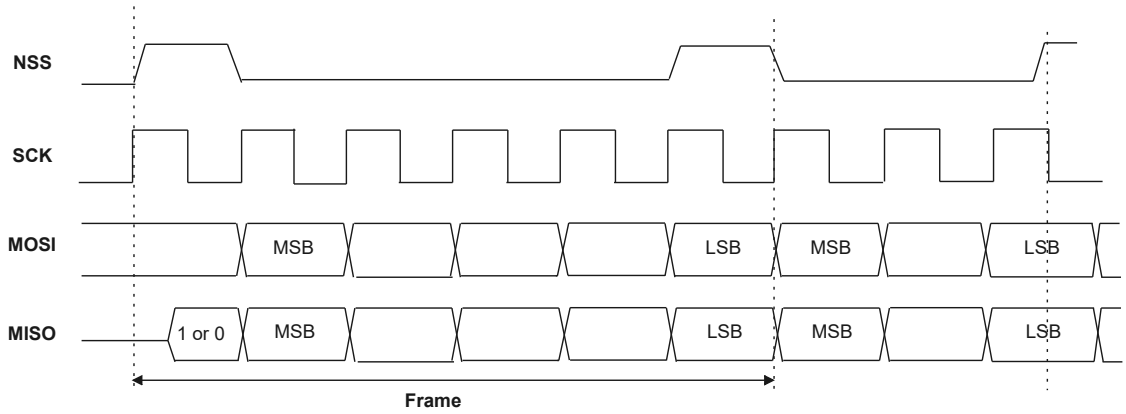


Figure 26-11 TI Mode Transfer

26.4.1.12 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 CKPH and CKPL bits in the SPIx_CTL0 register.

The SPI provides two types of CRC calculation methods, directly dependent on the chosen data frame format for transmission or reception: CRC8 for an 8-bit data frame and CRC16 for a 16-bit data frame.

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CTL0 register. This action resets the CRC registers (SPIx_TXCRC and SPIx_RXCRC). 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 SPIx_DATA. At the end of this last data transfer, the SPIx_TXCRC 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 SPIx_STAT 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 can be accomplished through the following procedure:

1. Program the CKPL, CKPH, LSBF, PRESCALER, and MSTR values.
2. Program the polynomial value in SPIx_CRCPOLY register (for CRC16_CCITT, CRCPOLY = 0x1021; for CRC16_USB, CRCPOLY = 0x8005).
3. Program the CRC initial value in SPIx_CRCCFG register (for CRC16_CCITT, CRC initial = 0x0; for CRC16_USB, CRC initial = 0xFFFF).
4. Program XOR output value in the SPIx_EXTCFG1 register (for CRC16_CCITT, XOR output by default is 0x0; for CRC16_USB, XOR output by default is 0xFFFF).
5. Enable the CRC calculation by setting CRCEN bit in the SPIx_CTL0 register. This also clears the SPIx_RXCRC and SPIx_TXCRC registers.
6. Enable SPI by setting the EN bit in the SPIx_CTL0 register.
7. Start the communication until all but one byte of half word has been transmitted or received.

a. In full duplex or transmitter-only mode, when transfers are managed by software, upon writing the final byte or half word to the Tx buffer, set the CRCNEXT bit in the SPIx_CTL0 register. This indicates that the CRC will be transmitted after transmission of the last byte.

b. In receive only mode, set the CRCNEXT bit just after the reception of the second to last data to prepare SPI to enter in CRC phase at the end of the reception of the last data.

8. After the transfer of the last byte or half word, the SPI enters the CRC transfer and check phase. In full duplex mode or receive only mode, the received CRC is compared to the SPIx_RXCRC value. If the value does not match, the CRCERR flag in SPIx_STAT register is set and an interrupt can be generated when the ERRIE bit in the SPIx_CTL1 register is set.

NOTE: When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is not automatic, software need to get into intervention to set CRCNEXT bit for CRC transfer.

26.4.1.13 Clock Dummy Function

This is one additional function for SPI mode and is started when SPI transfer completes if DUMMYEN bit is set. Under this mode, NSS will not be deasserted, DUMMYLEN[2:0] bits control how many SCK cycles to bypass outputting data on MOSI for master mode or MISO for slave mode and sampling input data on MISO for master mode or MOSI for slave mode. This function is useful when interfacing with LCD driver ICs, refer to below timing diagram for 3-line serial protocol for typical LCD driver IC.

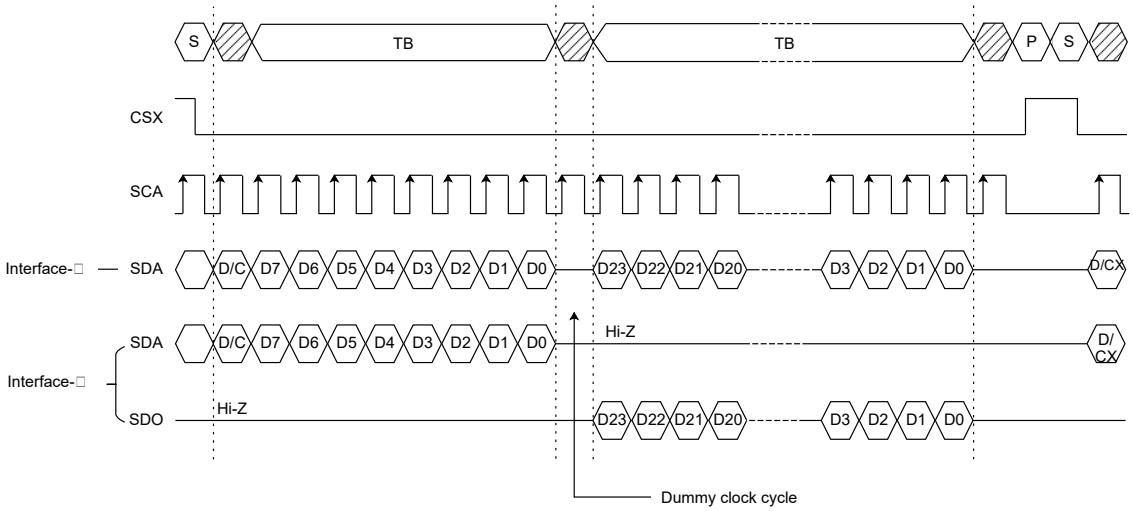


Figure 26-12 Clock Dummy Function Timing Sequence

26.4.1.14 Data Delay Function

This is one additional function for SPI mode which provides time delays before data transfer after NSS asserts. This is supported for all four SPI modes and TI mode and is only valid for master mode.

The delay time is inserted when DATDLYEN bit set and is configured with DATDLYLEN bits. Refer to Figure 26-13 for details.

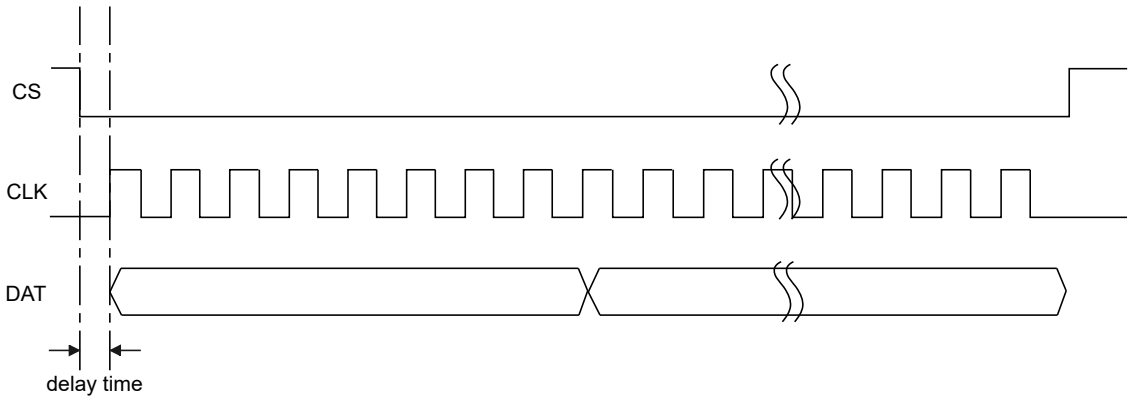
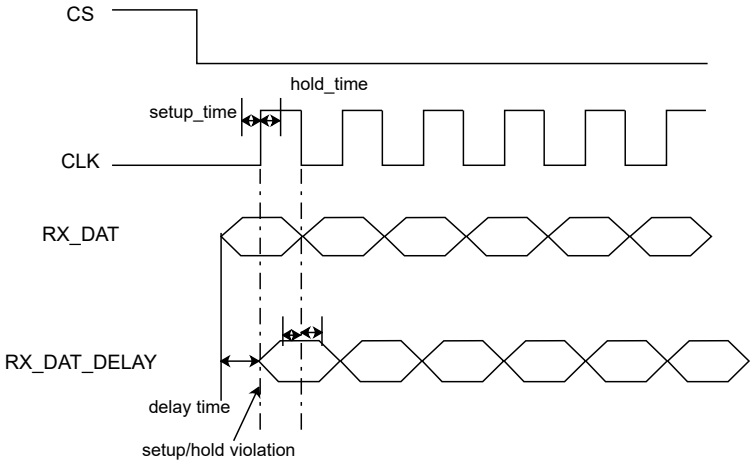


Figure 26-13 Data Delay Timing Sequence Diagram

26.4.1.15 SPI RX Sampling Edge Delay

This feature is provided to solve setup/hold time violation on SPI RX data line as shown in below figure and is valid only for CKPH = 0. When configuring SPI to work in SPI mode 0 with CKPH = 0 and CKPL = 0, data is normally sampled on the first rising clock edge, if due to RX_DAT line delay on PCB, it may cause setup/hold time violation and data sampling error. By configuring RXCLKREVTS = 1 in SPIx_EXTCFG0, the data sampling edge can be shifted back by one edge, enabling accurate data sampling.



26.4.2 I2S Functional Description

26.4.2.1 Block Diagram

Figure 26-14 shows the block diagram of Integrated Interchip Sound (I2S).

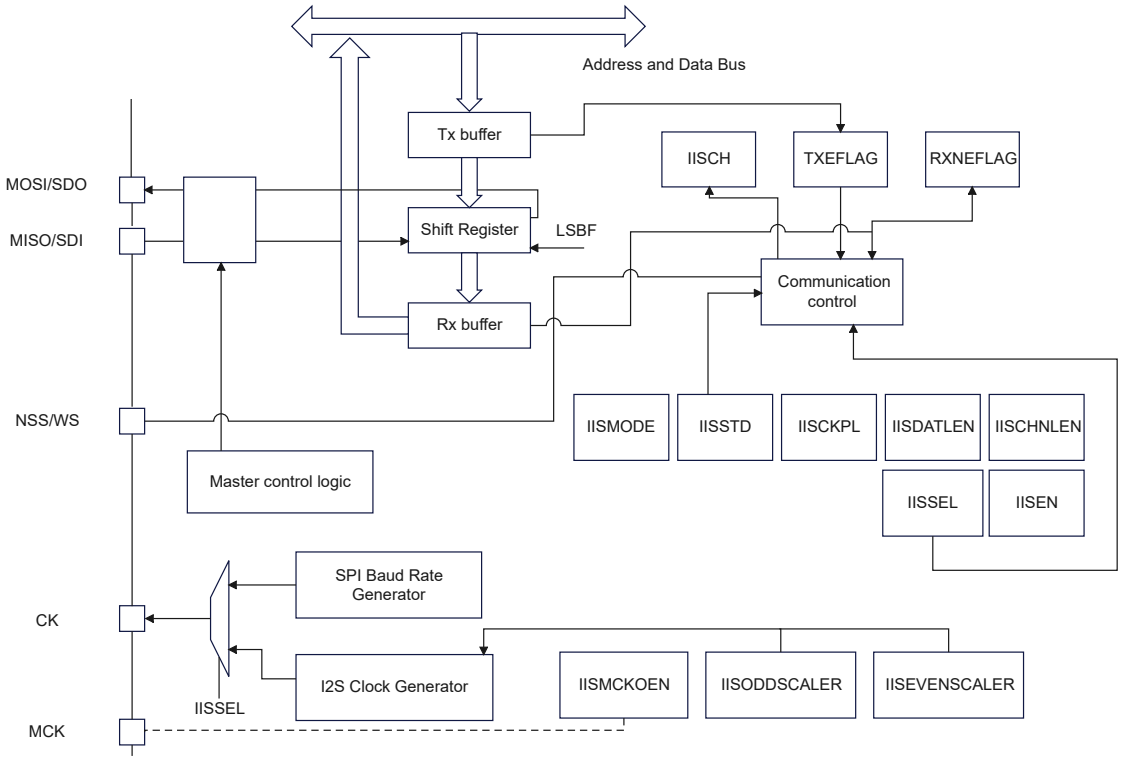


Figure 26-14 I2S Block Diagram

The SPI can function as an audio I2S interface when the I2S capability is enabled (by setting the IISSEL bit in the SPIx_IISCFG0 register). This interface mainly uses the same pins, flags and interrupts as the SPI.

The I2S shares four common pins with the SPI:

- SDO: Serial Data Output (mapped on the MOSI pin)
- SDI: Serial Data Input (mapped on the MISO pin)
- WS: Word Select (mapped on the NSSx 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 can 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 MCKOEN bit in the SPIx_IISCFG1 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 SPIx_IISCFG1, and the other one is a generic I2S configuration register SPIx_IISCFG0 (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPIx_CTL0 register, along with all CRC registers, are not used in I2S mode. Similarly, the NSSOESEL bit in the SPIx_CTL1 register and the MODEFAULT and CRCERR bits in the SPIx_STAT remain unused.

The I2S uses the same SPI register for data transfer (SPIx_DATA) for either 16-bit or 32-bit mode.

26.4.2.2 Supported Audio Protocol

The I2S bus has to handle only audio data, typically time-multiplexed across two channels: right and left. However, there is only one 32-bit register for transmission or reception. Consequently, it is up to the software to write into the data register the appropriate value corresponding to each channel side, or to read the data from the data register and to identify the corresponding channel by checking the IISCH bit in the SPIx_STAT register. When IISLRCKRFIRST = 0 (IISCH is not applicable for the PCM protocol), the left channel is transmitted first, followed by the right. Conversely, when IISLRCKRFIRST = 1 and IISLRCKSELEN = 1, the right channel is sent first, followed by the left.

There are five options for data and packet frames. The data can be sent in the following formats:

- 16-bit data packed in a 16-bit frame
- 16-bit data packed in a 32-bit frame
- 20-bit data packed in a 32-bit frame when IISDAT20BIT is set
- 24-bit data packed in a 32-bit frame
- 32-bit data packed in a 32-bit frame

When using 16-bit data extended on the 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).

When FIFO mode is disabled with FIFOBUFSEL = 0, the 20-bit, 24-bit, and 32-bit frames need two CPU read or write operations to/from the SPIx_DATA register or two DMA operations if the DMA is preferred for the application; when FIFO mode is enabled with FIFOBUFSEL = 1, 20-bit, 24-bit and 32-bit frames only need one CPU read/write operation or one DMA operation.

For all data formats and communication standards, the most significant bit is sent first (MSB first) when ENDIANSEL = 1, or the least significant bit is sent first (LSB first) when ENDIANSEL = 0.

The I2S interface supports four audio standards, configurable using the IISSTD[1:0] and IISPCMSYNCSEL bits in the SPIx_IISCFG0 register.

26.4.2.2.1 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. Normally it's left channel followed by right channel audio data, but it can be configured with IISLRCKRFIRST bit to choose right channel output first. Audio channel length can be configured to be 16-bit or 32-bit with IISCHNLEN bits, data length can be configured to be 16/20/24/32 bits with IISDATLEN and IISDAT20BIT bits.

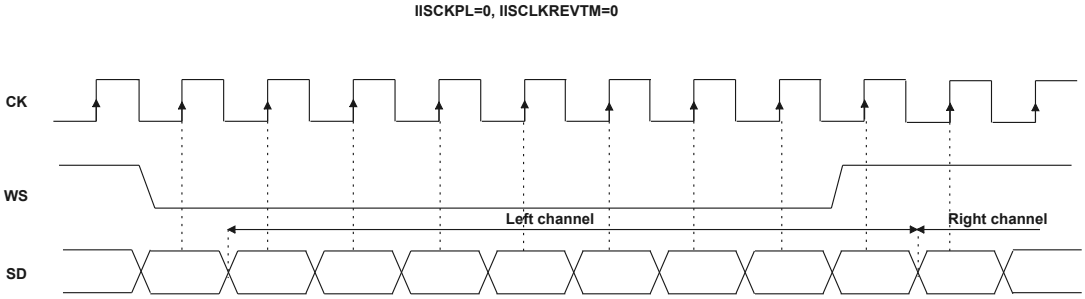


Figure 26-15 I2S Philips Protocol Waveforms (IISCKPL = 0, IISCLKREVTM = 0)

For I2S master mode, IISCLKREVTM bit can be configured to choose IIS output on inverted SCK edge, as shown in Figure 26-16.

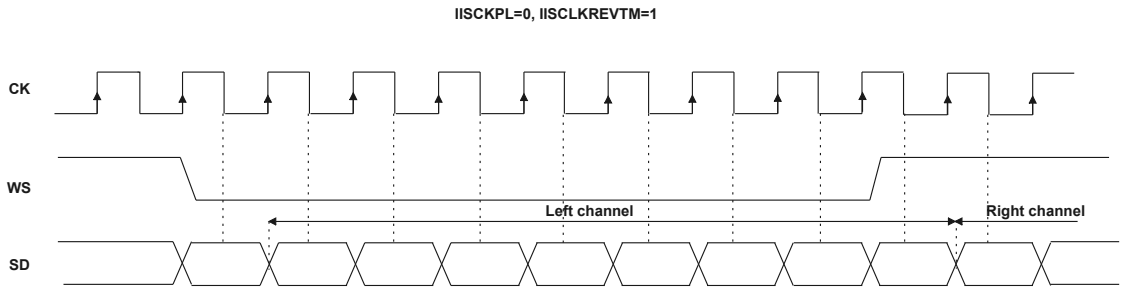


Figure 26-16 I2S Philips Protocol Waveforms (IISCKPL = 0, IISCLKREVTM = 1)

The IISCKPL bit can be configured to choose IIS clock steady state to be low level or high level, as shown in Figure 26-17.

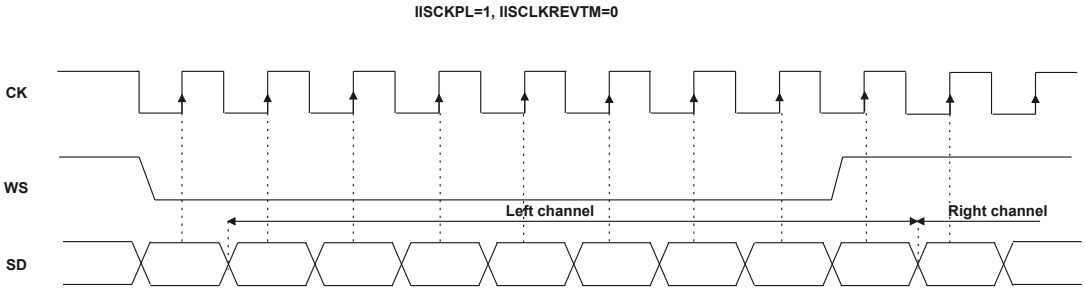


Figure 26-17 I2S Philips Protocol Waveforms (IISCKPL = 1, IISCLKREVTM = 0)

26.4.2.2.2 MSB Justified Standard

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSB bit. If data length is less than channel length, for example, 16-bit data length to transfer with 32-bit channel length, first 16-bit MSB data is output, then filled with 16-bit 0-forced data.

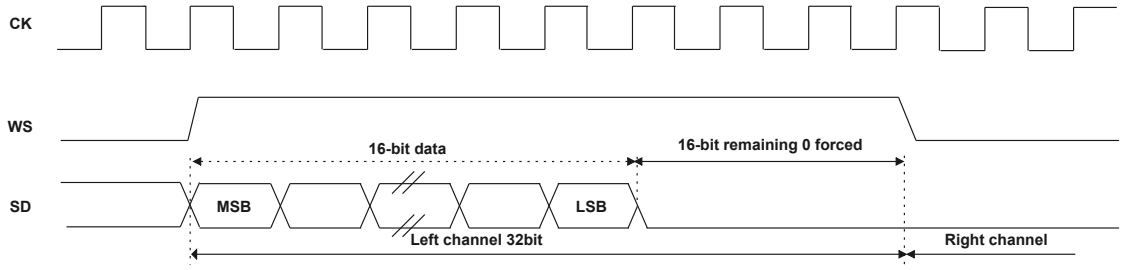


Figure 26-18 MSB Justified 16-bit Extended to 32-bit Packet Frame

26.4.2.2.3 LSB Justified Standard

This standard is similar to the MSB justified standard, except data is LSB aligned to WS, so for 16-bit data length transferred with 32-bit channel length, first 16-bit is 0-forced data, then followed with 16-bit data.

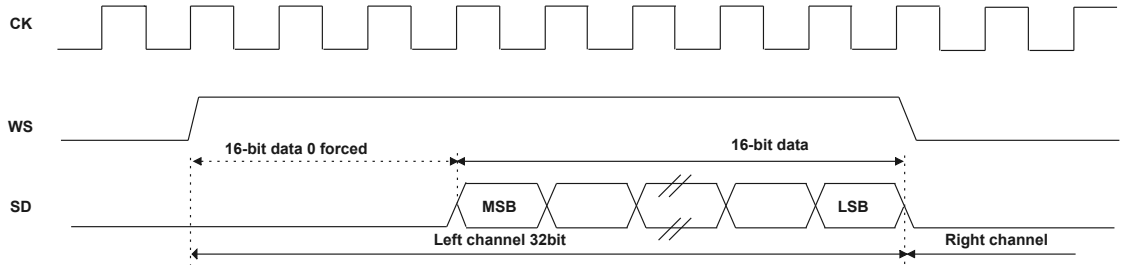


Figure 26-19 LSB Justified 16-bit Extended to 32-bit Packet Frame

26.4.2.2.4 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 IISPCMSYNCSEL bit in SPIx_IISCFG0 register. In PCM mode, the output signals (WS, SD) are sampled on the rising edge of CK signal. The input signals (WS, SD) are captured on the falling edge of CK. For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode. For short frame synchronization, the WS synchronization signal is only one cycle long.

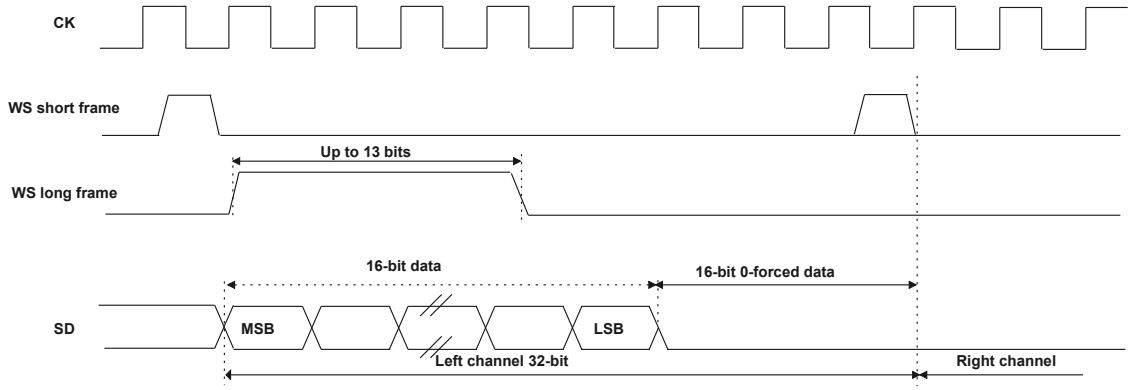


Figure 26-20 PCM Standard Waveforms (16-Bit Extended to 32-Bit Packet Frame)

26.4.2.3 Clock Generator

The I2S bit rate determines the data flow on the I2S data line and the I2S clock signal frequency.

I2S bit rate = number of bits per channel × number of channels × sampling audio frequency

For 16-bit audio, left and right channel, the I2S bit rate is calculated as follows:

I2S bit rate = 16 × 2 × f_S

It is: I2S bit rate = 32 x 2 x f_S if the packet length is 32-bit wide.

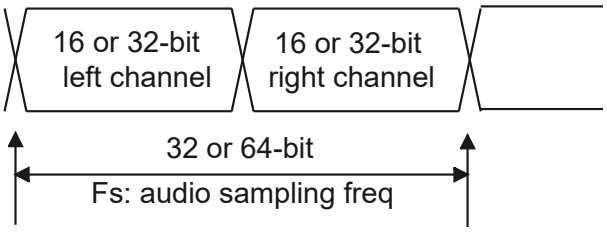


Figure 26-21 Audio Sampling Frequency Definition

When the master mode is configured, a specific action needs to be taken to properly program the linear divider to communicate with the desired audio frequency.

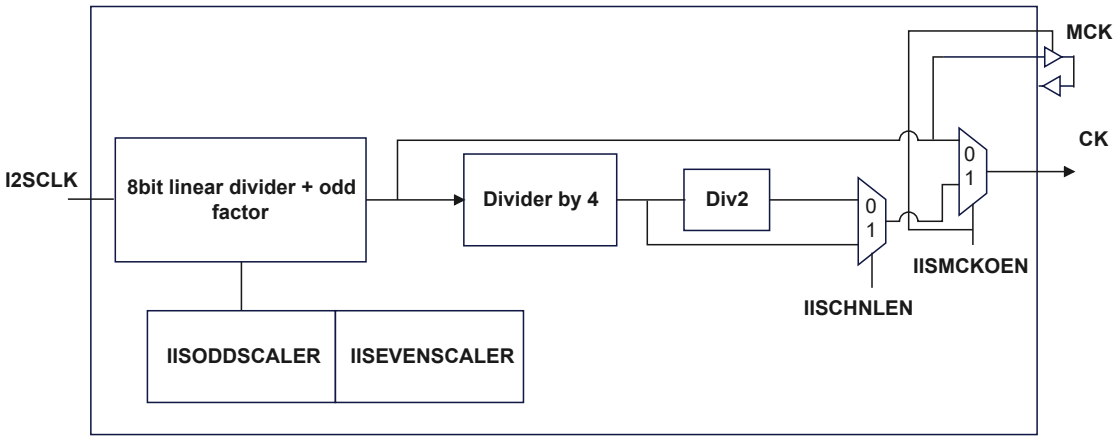


Figure 26-22 I2S Clock Generator Architecture

The I2SCLK clock is provided by the Reset and Clock Controller (RCC) of the product, the same as the APB clock used by the SPI/I2S block.

The audio sampling frequency may be 192 kHz, 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).

To reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

For I2S modes:

When the master clock is generated (IISMCKOEN in the SPIx_IISCFG1 register is set):

$$F_S = \frac{F_{I2SCLK}}{256 \times ((2 \times IISEVENSCALER) + IISODDSCALER)} \quad (11)$$

When the master clock is disabled (IISMCKOEN bit cleared):

$$F_S = \frac{F_{I2SCLK}}{32 \times (IISCHNLEN + 1) \times ((2 \times IISEVENSCALER) + IISODDSCALER)} \quad (12)$$

IISCHNLEN = 0 when the channel frame is 16-bit wide and, IISCHNLEN = 1 when the channel frame is 32-bit wide.

For PCM modes:

When the master clock is generated (IISMCKOEN in the SPIx_IISCFG1 register is set):

$$F_S = \frac{F_{I2SCLK}}{128 \times ((2 \times IISEVENSCALER) + IISODDSCALER)} \quad (13)$$

When the master clock is disabled (IISMCKOEN bit cleared):

$$F_S = \frac{F_{I2SCLK}}{16 \times (IISCHNLEN + 1) \times ((2 \times IISEVENSCALER) + IISODDSCALER)} \quad (14)$$

IISCHNLEN = 0 when the channel frame is 16-bit wide and, IISCHNLEN = 1 when the channel frame is 32-bit wide.

NOTE:
- F_S is the audio sampling frequency, and F_{I2SCLK} is the frequency of the kernel clock provided to the SPI/I2S block.
- IISEVENSCALER must be higher than 1.

Table 26-2 and Table 26-3 provide example precision values for different prescaler configurations under SYSCLK = 156 MHz (with IHS = 8 MHz, PLLINDIV = 0 (DIV1), PLLFBDIV = 39, PLLODIV = 0 (DIV2)).

Table 26-2 Audio-frequency Precision for I2S Mode and MCLK Disabled

SYSCLK (MHz)	Data Length	Target Fs (kHz) IIS_FS	Prescaler (1000xSYSCLK)/ (IIS_FS*64)	IISEVENSCALER	IISODDSCALER	MCLK	Error (%)
156	32	192	13	6	1	No	-2.344
156	32	96	25	12	1	No	1.563
156	32	48	51	25	1	No	-0.429
156	32	44.1	55	27	1	No	0.495
156	32	32	76	38	0	No	0.226
156	32	22.05	111	55	1	No	-0.411

Table 26-3 Audio-frequency Precision for I2S Mode and MCLK Enabled

SYSCLK (MHz)	Data Length	Target Fs (kHz) IIS_FS	Prescaler (1000xSYSCLK)/ (IIS_FS*64*4)	IISEVENSCALER	IISODDSCALER	MCLK	Error (%)
156	32	192	3	1	1	Yes	5.794
156	32	96	6	3	0	Yes	5.794
156	32	48	13	6	1	Yes	-2.344
156	32	44.1	14	7	0	Yes	-1.3
156	32	32	19	9	1	Yes	0.226
156	32	22.05	28	14	0	Yes	-1.3

26.4.2.4 I2S Master Mode

The I2S can be configured in master mode. The serial clock is generated on the CK pin and the Word Select signal (WS). The Master Clock (MCK) may be output or not, controlled by the IISMCKOEN bit in the SPIx_IISCFG1 register.

26.4.2.4.1 Procedure

1. Select the IISEVENSCALER[7:0] bits in the SPIx_IISCFG1 register to define the serial clock baud rate to reach the proper audio sample frequency. The IISODDSCALER bit in the SPIx_IISCFG1 register also has to be defined.
2. Select the IISCKPL bit to define the steady level for the communication clock. Set the IISMCKOEN bit in the SPIx_IISCFG1 register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the IISEVENSCALER and IISODDSCALER values should be computed depending on the state of the MCK output; for more details, refer to Clock Generator).
3. Set the IISSEL bit in the SPIx_IISCFG0 register to activate the I2S functions and choose the I2S standard through the IISSTD[1:0] and IISPCMSYNCSEL bits, the data length through the IISDATLEN[1:0] bits and the number of bits per channel by configuring the IISCHNLEN bit. Select the I2S master mode and direction (Transmitter or Receiver) through the IISMODE[1:0] bits in the SPIx_IISCFG0 register.

4. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CTL1 register.
5. The IISEN bit in the SPIx_IISCFG0 register must be set.

WS and CK are configured in output mode. MCK is also output if the IISMCKOEN bit in SPIx_IISCFG1 is set.

26.4.2.4.2 Transmission Sequence

The transmission sequence begins when data is written into the Tx buffer.

Suppose the first data written into the Tx buffer corresponds to the left channel data. The TXEFLAG is set when the data are transferred from the Tx buffer to the shift register, and data corresponding to the right channel must be written into the Tx buffer. The IISCH flag indicates which channel is to be transmitted. It has a meaning when the TXEFLAG flag is set because the IISCH flag is updated when TXEFLAG 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 TXEFLAG flag is set after each transfer from the Tx buffer to the shift register, and an interrupt is generated if the TXFEIE bit in the SPIx_CTL1 register is set.

For more details about the write operations depending on the I2S standard mode selected, refer to Supported Audio Protocol.

To ensure a continuous audio data transmission, it is mandatory to write the SPIx_DATA register with the next data to transmit before the end of the current transmission.

To switch off the I2S by clearing IISEN, it is mandatory to wait for TXEFLAG = 1 and BUSY = 0.

26.4.2.4.3 Reception Sequence

The operating mode is the same as for transmission mode except for the point 3 (refer to the procedure described in I2S Master Mode), where the configuration should set the master reception mode through the IISMODE[1:0] bits.

Each time the Rx buffer is full, the RXNEFLAG flag is set and an interrupt is generated if the RXFNEIE bit is set in SPIx_CTL1 register.

Clearing the RXNEFLAG bit is performed by reading the SPIx_DATA register.

IISCH 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 Supported Audio Protocol.

If data are received while the previously received data have not been read yet, an overrun is generated and the RXFOVERFLOW flag is set. If the ERRIE bit is set in the SPIx_CTL1 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 (IISDATLEN = 00 and IISCHNLEN = 1) using the LSB justified mode (IISSTD = 10)

  1. Wait for the second to last RXNEFLAG = 1 (n – 1)
  2. Then wait 17 I2S clock cycles (using a software loop)
  3. Disable the I2S (IISEN = 0)

- 16-bit data length extended on 32-bit channel length (IISDATLEN = 00 and IISCHNLEN = 1) in MSB justified, I2S or PCM modes (IISSTD = 00, IISSTD = 01 or IISSTD = 11, respectively).

  1. Wait for the last RXNEFLAG.
  2. Then wait 1 I2S clock cycle (using a software loop).
  3. Disable the I2S (IISEN = 0).
- For all other combinations of IISDATLEN and IISCHNLEN, whatever the audio mode selected through the IISSTD bits, carry out the following sequence to switch off the I2S:

  1. Wait for the second to last RXNEFLAG = 1 (n – 1).
  2. Then wait one I2S clock cycle (using a software loop).
  3. Disable the I2S (IISEN = 0).

NOTE: The BUSY flag is kept low during transfers.

26.4.2.5 I2S Slave Mode

For the slave configuration, 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 IISSEL bit in the SPIx_IISCFG0 register to select I2S mode and choose the I2S standard through the IISSTD[1:0] bits, the data length through the IISDATLEN[1:0] bits and the number of bits per channel for the frame configuring the IISCHNLEN bit. Select also the mode (transmission or reception) for the slave through the IISMODE[1:0] bits in SPIx_IISCFG0 register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx_CTL1 register.
3. The IISEN bit in SPIx_IISCFG0 register must be set.

26.4.2.5.1 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 TXEFLAG flag is then set to request the right channel data to be written into the I2S data register.

The IISCH flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, IISCH 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 IISEN 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 TXEFLAG flag

is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXFEIE bit in the SPIx_CTL1 register is set.

NOTE: The TXEFLAG 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 Supported Audio Protocol.

To secure a continuous audio data transmission, it is mandatory to write the SPIx_DATA 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 SPIx_DATA 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 SPIx_CTL1 register, an interrupt is generated when the TXFUNDERFLOW flag in the SPIx_STAT 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 IISEN bit, it is mandatory to wait for TXEFLAG = 1 and BUSY = 0.

26.4.2.5.2 Reception Sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in I2S Slave Mode), where the configuration should set the master reception mode using the IISMODE[1:0] bits in the SPIx_IISCFG0 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 RXNEFLAG flag in the SPIx_STAT register is set and an interrupt is generated if the RXFNEIE bit is set in the SPIx_CTL1 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 IISCH flag is updated each time data are received to be read from the SPIx_DATA register. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNEFLAG bit is performed by reading the SPIx_DATA register.

For more details about the read operations depending the I2S standard mode selected, refer to Supported Audio Protocol.

If data are received while the preceding received data have not yet been read, an overrun is generated and the RXFOVERFLOW flag is set. If the bit ERRIE is set in the SPIx_CTL1 register, an interrupt is generated to indicate the error.

To switch off the I2S in reception mode, IISEN has to be cleared immediately after receiving the last RXNEFLAG = 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.

26.4.2.6 DMA Features

In I2S mode, the DMA works in exactly the same way as it does in SPI mode. There is no difference except that the CRC feature is not available in I2S mode since there is no data transfer protection system.

26.5 Status, Errors and Interrupts

26.5.1 SPI Status, Errors and Interrupts

26.5.1.1 SPI Status Flags

The following status flags are provided for the application to completely monitor the state of the SPI bus.

- **Rx FIFO Empty Flag (RXFEMPTYFLAG)**

  The RXFEMPTYFLAG bit is set when RX FIFO becomes empty, with no entries for either FIFO mode (FIFOBUFSEL = 1) or BUFFER mode (FIFOBUFSEL = 0).

- **Rx FIFO Full Flag (RXFFULLFLAG)**

  The RXFFULLFLAG bit is set when RX FIFO is full (with 8 entries in FIFO), this bit is valid only for FIFO mode (when FIFOBUFSEL = 1). An interrupt can be generated if the RXFFIE bit in SPIx_CTL1 register is set. The RXFFULLFLAG is cleared by hardware automatically when the above conditions are no longer true.

- **Rx Buffer not Empty Flag (RXNEFLAG)**

  The RXNEFLAG bit is set when RX FIFO entries reaches the threshold set by RXFAFLVL in FIFO mode (FIFOBUFSEL = 1) or FIFO entry is 1 in BUFFER mode (FIFOBUFSEL = 0). An interrupt can be generated if the RXFNEIE bit in the SPIx_CTL1 register is set. The RXNEFLAG is cleared by hardware automatically when the above conditions are no longer true.

- **Tx Buffer Empty Flag (TXEFLAG)**

  The TXEFLAG flag is set when transmission TXFIFO becomes empty, with no entries for both FIFO mode (FIFOBUFSEL = 1) and BUFFER mode (FIFOBUFSEL = 0). An interrupt can be generated if the TXFEIE bit in the SPIx_CTL1 register is set. This bit is cleared automatically when the TXFIFO level has at least one entry.

- **Tx FIFO Full Flag (TXFFULLFLAG)**

  The TXFFULLFLAG flag is set when TXFIFO is full (with 8 FIFO entries), this bit is only valid in FIFO mode (FIFOBUFSEL = 1).

- **Tx FIFO Almost Empty Flag (TXFAEMPTYFLAG)**

  The TXFAEMPTYFLAG bit is set when transmission TXFIFO has enough space to store data to send. This flag is linked to the TXFIFO level, it goes high and stays high until the TXFIFO level is lower than the TXFIFO threshold set by TXFAELVL bits. An interrupt can be generated if the TXFAEIE bit in the SPIx_CTL1 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than TXFIFO threshold.

  The following table shows SPI status flag relationship with data entries for Buffer mode or FIFO mode.

SPI Status Flags	Buffer Mode (Data Entries)	FIFO Mode (Data Entries)
RXNEFLAG	1	RXFIFO threshold (RXFAFLVL)
RXFFULLFLAG	N/A	8
RXFEMPTYFLAG	0	0
TXEFLAG	0	0

SPI Status Flags	Buffer Mode (Data Entries)	FIFO Mode (Data Entries)
TXFAEMPTYFLAG	N/A	TXFIFO threshold (TXFAELVL)
TXFFULLFLAG	N/A	8

- **Data Transfer Flag (DATATRANS)**

  The DATATRANS flag is set when SPI data is being transferred, the difference between DATATRANS and BUSY flag is DATATRANS remains high during data transfer and will not be set to ‘0’ for one clock cycle between data transfer.

- **Busy Flag (BUSY)**

  The BUSY flag is set and cleared by hardware (writing to this flag has no effect).

  When BUSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

  The BUSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

  The BUSY flag is also useful for preventing write collisions in a multimaster system. The BUSY flag is cleared under any one of the following conditions:

  – When the SPI is correctly disabled
  – When a fault is detected in master mode (MODEFAULT bit set to 1)
  – In master mode, when it finishes a data transmission and no new data is ready to be sent
  – In slave mode, when the BUSY flag is set to '0' for at least one SPI clock cycle between each data transfer

26.5.1.2 SPI Error Flags

An SPI interrupt is generated if one of the following error flags is set and the interrupt is enabled by setting the ERRIE bit.

- **Overrun Flag (RXFOVERFLOW)**

  An overrun condition occurs when data is received by a master or slave, and the RXFIFO or Rx Buffer does not have enough space to store this received data. This can happen if the software or the DMA does not have enough time to read the previously received data (stored in the RXFIFO/Rx Buffer) or when space for data storage is limited.

  When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO/Rx Buffer. The newly received value is discarded, and all data transmitted subsequently is lost. Clearing the RXFOVERFLOW bit is done by a read access to the SPIx_DATA register followed by a read access to the SPIx_STAT register.

- **Mode Fault (MODEFAULT)**

  Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SWNSS bit in NSS software mode) pulled low. This automatically sets the MODEFAULT bit. Master mode fault affects the SPI interface in the following ways:

  – The MODEFAULT bit is set, and an SPI interrupt is generated if the ERRIE bit is set.
  – The EN 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 MODEFAULT bit:

1. Make a read or write access to the SPIx_STAT register while the MODEFAULT bit is set.
2. Write to the SPIx_CTL0 register.

- **CRC Error (CRCERR)**

  This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx_CTL0 register is set. The CRCERR flag in the SPIx_STAT register is set if the value received in the shift register does not match the receiver SPIx_RXCRC value. The flag is cleared by the software.

- **TI Mode Frame Format Error (FRMERR)**

  A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI operates in slave mode and is configured to conform to the TI mode protocol. When this error occurs, the FRMERR flag is set in the SPIx_STAT register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer.

  The FRMERR flag is cleared when the SPIx_STAT register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection.

26.5.1.3 SPI Interrupts

During SPI communication an interrupt can be generated by the following events:

- Transmit buffer or TXFIFO empty
- Receive RXFIFO full
- Transmit TXFIFO ready to be loaded
- Data received in Receive buffer or RXFIFO
- Master mode fault
- Overrun error
- TI frame format error
- CRC protocol error

Interrupts can be enabled and disabled separately.

Table 26-4 SPI Interrupt Requests

Interrupt Event	Event Flag	Enable Control Bit
Transmit buffer or TXFIFO empty	TXEFLAG	TXFEIE
Receive RXFIFO full	RXFFULLFLAG	RXFFIE
Transmit TXFIFO ready to be loaded	TXFAEMPTYFLAG	TXFAEIE
Data received in Rx buffer or RXFIFO	RXNEFLAG	RXFNEIE
Master Mode fault event	MODEFAULT	ERRIE
Overrun error	RXFOVERFLOW	
TI frame format error	FRMERR	
CRC protocol error	CRCERR	

26.5.2 I2S Status, Errors and Interrupts

26.5.2.1 I2S Status Flags

The following status flags are provided for the application to monitor the state of the I2S bus fully.

- **Busy Flag (BUSY)**

  The BUSY 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 BUSY is set, it indicates that the I2S is busy communicating. There is one exception in master receive mode (IISMODE = 11), where the BUSY flag is kept low during reception.

  The BUSY 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 BUSY flag is set when a transfer starts, except when the I2S is in master receiver mode. The BUSY 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 BUSY flag is kept high during all the transfers
  – In slave mode, the BUSY flag goes low for one I2S clock cycle between each transfer

  NOTE: Do not use the BUSY flag to handle each data transmission or reception. It is better to use the TXEFLAG and RXNEFLAG bits instead.

- **TX Buffer Empty Flag (TXEFLAG)**

  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 TXEFLAG bit is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I2S is disabled (IISEN bit is reset).

- **RX Buffer not Empty (RXNEFLAG)**

  When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when the SPIx_DATA register is read.

- **Channel Side Flag (IISCH)**

  In transmission mode, this flag is refreshed when TXEFLAG 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 SPIx_DATA. It indicates from which channel side data have been received. Note that in case of an error (like RXFOVERFLOW), 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 RXFOVERFLOW or TXFUNDERFLOW flag in the SPIx_STAT is set, and the ERRIE bit in SPIx_CTL1 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPIx_STAT status register (once the interrupt source has been cleared).

26.5.2.2 I2S Error Flags

There are three error flags for the I2S.

- **Underrun Flag (TXFUNDERFLOW)**

  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 SPIx_DATA. It is available when the IISSEL bit in the SPIx_IISCFG0 register is set. An interrupt may be generated if the ERRIE bit in the SPIx_CTL1 register is set.

  The TXFUNDERFLOW bit is cleared by a read operation on the SPIx_STAT register.

- **Overrun Flag (RXFOVERFLOW)**

  This flag is set when data are received, and the previous data have not yet been read from the SPIx_DATA register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx_CTL1 register.

  In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx_DATA register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

  Clearing the RXFOVERFLOW bit is done by a read operation on the SPIx_DATA register followed by a read access to the SPIx_STAT register.

- **Frame Error Flag (FRMERR**)

  This flag can be set by hardware only if the I2S is configured in slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the synchronization is lost, the following steps are required to recover from this state and resynchronize the external master device with the I2S slave device:

  1. Disable the I2S.
  2. Enable it again when the correct level is detected on the WS line (WS line is high in I2S mode or low for MSB- or LSB-justified or PCM modes.

  Desynchronization between master and slave devices may be due to a noisy environment on the CK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRMERR) is cleared by software when the status register is read.

26.5.2.3 I2S Interrupts

Table 26-5 lists the I2S interrupts.

Table 26-5 I2S Interrupt Requests

Interrupt Event	Event Flag	Enable Control Bit
Transmit buffer empty flag	TXEFLAG	TXFEIE
Receive buffer not empty flag	RXNEFLAG	RXFNEIE
Overrun error	RXFOVERFLOW	ERRIE
Underrun error	TXFUNDERFLOW	
Frame error flag	FRMERR	

26.6 Registers

26.6.1 Register Address Map

Offset	Register Name	Register Description
0x0000	SPI_CTL0	SPI control register 0
0x0004	SPI_CTL1	SPI control register 1
0x0008	SPI_STAT	SPI status register
0x000c	SPI_DATA	SPI data register
0x0010	SPI_CRCPOLY	SPI CRC polynomial register
0x0014	SPI_RXCRC	SPI RX CRC register
0x0018	SPI_TXCRC	SPI TX CRC register
0x001c	SPI_IISCFG0	IIS configuration register 0
0x0020	SPI_IISCFG1	IIS configuration register 1
0x0024	SPI_EXTCFG0	SPI extra configuration register 0
0x0028	SPI_FIFOCFG	SPI FIFO configuration register
0x002c	SPI_CRCCFG	SPI CRC initial configuration register
0x0030	SPI_EXTCFG1	SPI extra configuration register 1
0x0034	SPI_EXTCFG2	SPI extra configuration register 2
0x0038	SPI_EXTCFG3	SPI extra configuration register 3

26.6.2 Register Field Details

26.6.2.1 SPI_CTL0

0x0000			SPI control register 0									SPI_CTL0				
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BIDEN	BIDOEN	CRCEN	CRCNEXT	FRMFORMAT	RXONLY	SWNSSEN	SWNSS	LSBF	EN	PRESCALER			MSTR	CKPL	CKPH
Type	RW	RW	RW	RS	RW	RW	RW	RW	RW	RW	RW			RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-6 SPI Control Register 0 Description

Field	Name	Description
31:16	Reserved	Reserved
15	BIDEN	Bidirectional data mode enable This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active. 0: 2-line unidirectional data mode selected 1: 1-line bidirectional data mode selected **NOTE:** This bit is not used in IIS mode.
14	BIDOEN	Output enable in bidirectional mode This bit, combined with the BIDEN bit, selects the direction of transfer in bidirectional mode. 0: Output disabled (receive-only mode)

Field	Name	Description
		1: Output enabled (transmit-only mode) **NOTE:** In master mode, the MOSI pin is used, and in slave mode, the MISO pin is used. This bit is not used in IIS mode.
13	CRCEN	Hardware CRC calculation enable 0: CRC calculation disabled 1: CRC calculation enabled **NOTE:** This bit should be written only when SPI is disabled (EN = '0') for correct operation. This bit is not used in IIS mode.
12	CRCNEXT	Transmit CRC next 0: Next transmit value is from Tx buffer. 1: Next transmit value is from Tx CRC register. **NOTE:** This bit has to be written as soon as the last data is written in the DATA register. This bit is not used in IIS mode.
11	FRMFORMAT	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 (EN = '0') for correct operation. It is not used in IIS mode.
10	RXONLY	Receive only This bit combined with the BIDEN bit selects the direction of transfer in 2-line unidirectional mode. This bit is also useful in a multi slave 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) **NOTE:** This bit is not used in IIS mode.
9	SWNSSEN	Software slave select enable

Field	Name	Description
		When the SWNSS bit is set, the NSS pin input is replaced with the value from the SWNSS bit. 0: Software slave management disabled 1: Software slave management enabled **NOTE:** This bit is not used in IIS mode.
8	SWNSS	Software slave select This bit only affects when the SWNSS bit is set. The value of this bit is forced onto the NSS pin, and the IO value of the NSS pin is ignored. **NOTE:** This bit is not used in IIS mode.
7	LSBF	Frame format 0: MSB transmitted first 1: LSB transmitted first **NOTE:** This bit should not be changed when communication is ongoing. It is not used in IIS mode.
6	EN	SPI enable 0: Peripheral disabled 1: Peripheral enabled **NOTE:** This bit is not used in IIS mode.
5:3	PRESCALER	Baud rate control 000: $f_{PCLK}/2$ 001: $f_{PCLK}/4$ 010: $f_{PCLK}/8$ 011: $f_{PCLK}/16$ 100: $f_{PCLK}/32$ 101: $f_{PCLK}/64$

Field	Name	Description
		110: $f_{PCLK}/128$ 111: $f_{PCLK}/256$ **NOTE:** These bits should not be changed when communication is ongoing. They are not used in IIS mode.
2	MSTR	Master selection 0: Slave configuration 1: Master configuration **NOTE:** This bit should not be changed when communication is ongoing. This bit is not used in IIS mode.
1	CKPL	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. It is not used in IIS mode.
0	CKPH	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. It is not used in IIS mode.

26.6.2.2 SPI_CTL1

0x0004			SPI control register 1											SPI_CTL1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RXMASK	RXFFIE	TXFAEIE	Reserved					TXFEIE	RXFNEIE	ERRIE	TIMODE	NSSMODE	NSSOESEL	DMATXEN	DMARXEN
Type	RW	RW	RW	RO					RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 26-7 SPI Control Register 1 Description

Field	Name	Description
31:16	Reserved	Reserved
15	RXMASK	SPI simplex mode Rx mask 0: Data will be received as usual in SPI Tx simplex mode. 1: Data will not be received in SPI Tx simplex mode.
14	RXFFIE	Rx FIFO full interrupt enable 0: Disable RXF interrupt 1: Enable RXF interrupt Only used in FIFO mode
13	TXFAEIE	Tx FIFO almost empty interrupt enable 0: Disable TXAE interrupt 1: Enable TXAE interrupt Only used in FIFO mode

Field	Name	Description
12:8	Reserved	Reserved
7	TXFEIE	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	RXFNEIE	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	ERRIE	Error interrupt enable This bit controls the generation of an interrupt when an error condition occurs (CRCERR, RXFOVERFLOW, MODEFAULT, FRMERR in SPI mode, FRE at TI mode, and TXFUNDERFLOW, RXFOVERFLOW, and FRMERR in IIS mode). 0: Error interrupt is masked 1: Error interrupt is enabled
4	TIMODE	SPI TI mode enable 0: SPI TI mode disable 1: SPI TI mode enable
3	NSSMODE	SPI NSS mode enable 0: SPI NSS mode disable 1: SPI NSS mode enable
2	NSSOESEL	NSS output enable 0: NSS output is disabled in master mode and the cell can work in multimaster configuration. 1: NSS output is enabled in master mode and when the cell is enabled. The cell cannot work in a multimaster environment.

Field	Name	Description
1	DMATXEN	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	DMARXEN	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

26.6.2.3 SPI_STAT

0x0008			SPI status register											SPI_STAT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RXFEMPTYFLAG	RXFFULLFLAG	TXFAEMPTYFLAG	TXFFULLFLAG	Reserved		DATATRANS	FRMERR	BUSY	RXFOVERFLOW	MODEFAULT	CRCERR	TXFUNDERFLOW	IISCH	TXEFLAG	RXNEFLAG
Type	RO	RO	RO	RO	RO		RO	RO	RO	WC	WC	WC	WC	RO	RO	RO
Reset	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 26-8 SPI Status Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	RXFEMPTYFLAG	Receive FIFO empty 0: Rx FIFO not empty 1: Rx FIFO empty
14	RXFFULLFLAG	Receive FIFO full 0: Rx FIFO not full 1: Rx FIFO full
13	TXFAEMPTYFLAG	Transmit FIFO almost empty 0: Tx FIFO not almost empty 1: Tx FIFO almost empty

Field	Name	Description
12	TXFFULLFLAG	Transmit FIFO full 0: Tx FIFO not full 1: Tx FIFO full
11:10	Reserved	Reserved
9	DATATRANS	SPI data transfer flag 0: SPI is in idle or waiting state. 1: SPI data is being transferred.
8	FRMERR	Frame format error This flag is set by hardware and reset when STAT is read by software. 0: No frame format error 1: A frame format error occurred
7	BUSY	Busy flag 0: SPI or IIS not busy 1: SPI or IIS is busy in communication, or Tx buffer is not empty This flag is set and cleared by hardware.
6	RXFOVERFLOW	Overflow flag 0: No overflow occurred 1: Overflow occurred This flag is set by hardware and reset by a software sequence.
5	MODEFAULT	Mode fault 0: No mode fault occurred 1: Mode fault occurred This flag is set by hardware and reset by a software sequence. **NOTE:** This bit is not used in IIS mode.

Field	Name	Description
4	CRCERR	CRC error flag 0: CRC value received matches the RXCRCR value 1: CRC value received does not match the RXCRCR value This flag is set by hardware and cleared by software writing 0. **NOTE:** This bit is not used in IIS mode.
3	TXFUNDERFLOW	Underflow flag 0: No underflow occurred 1: Underflow occurred This flag is set by hardware and reset by a software sequence.
2	IISCH	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	TXEFLAG	Transmit buffer empty 0: Tx buffer not empty 1: Tx buffer empty
0	RXNEFLAG	Receive buffer not empty 0: Rx buffer empty 1: Rx buffer not empty

26.6.2.4 SPI_DATA

0x000c			SPI data register											SPI_DATA		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DAT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DAT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-9 SPI Data Register Description

Field	Name	Description
31:0	DAT	Data received or to be transmitted. The data register is split into two buffers: One for writing (Transmit buffer) and another 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. Please note these points are relevant in SPI mode: Depending on the data frame format selection bit (FRMFORMAT in SPI_CTL0 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 (DATA[7:0]) is used for transmission or reception. When in reception mode, the MSB of the register (DATA[15:8]) is forced to 0.

Field	Name	Description
		For a 16-bit data frame, the buffers are 16-bit, and the entire register, DATA[15:0], is used for transmission or reception.

26.6.2.5 SPI_CRCPOLY

0x0010			SPI CRC polynomial register											SPI_CRCPOLY		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CRCPOLY															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Table 26-10 SPI CRC Polynomial Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CRCPOLY	CRC polynomial register This register contains the polynomial for the CRC calculation. The CRC polynomial (0x0007) is the reset value of this register. Another polynomial can be configured as required.

26.6.2.6 SPI_RXCRC

0x0014			SPI RX CRC register											SPI_RXCRC		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CRCRXRSLT															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-11 SPI RX CRC Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CRCRXRSLT	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 the CTL0 register is written to 1. The CRC is calculated serially using the polynomial programmed in the CRCPOLY register. Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (FRMFORMAT bit in the CTL0 is cleared). CRC calculation is done based on any CRC8 standard. The entire 16-bit of this register is considered when a 16-bit CRC frame format is selected (FRMFORMAT bit in the CTL0 register is set). CRC calculation is done based on any CRC16 standard.

Field	Name	Description
		NOTE: A read to this register when the BUSY flag is set could return an incorrect value. These bits are not used in IIS mode.

26.6.2.7 SPI_TXCRC

0x0018			SPI TX CRC register											SPI_TXCRC		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CRCTXRSLT															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-12 SPI TX CRC Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CRCTXRSLT	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 CTL0 is written to 1. The CRC is calculated serially using the polynomial programmed in the CRCPOLY register. Only the 8 LSB bits are considered when the CRC frame format is set to be 8-bit length (FRMFORMAT bit in the CTL0 is cleared). CRC calculation is done based on any CRC8 standard. The entire 16-bit of this register is considered when a 16-bit CRC frame format is selected (FRMFORMAT bit in the CTL0 register is set). CRC calculation is done based on any CRC16 standard. **NOTE:** A read to this register when the BUSY flag is set could return an incorrect value. These bits are not used in IIS mode.

26.6.2.8 SPI_IISCFG0

0x001c			IIS configuration register 0											SPI_IISCFG0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				IISSEL	IISEN	IISMODE		IISPCMSYNCSEL	Reserved	IISSTD		IISCKPL	IISDATLEN		IISCHNLEN
Type	RO				RW	RW	RW		RW	RO	RW		RW	RW		RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-13 IIS Configuration Register 0 Description

Field	Name	Description
31:12	Reserved	Reserved
11	IISSEL	IIS mode selection 0: SPI mode is selected 1: IIS mode is selected **NOTE:** This bit should be configured when the SPI or IIS is disabled.
10	IISEN	IIS Enable 0: IIS peripheral is disabled 1: IIS peripheral is enabled

Field	Name	Description
		NOTE: This bit is not used in SPI mode.
9:8	IISMODE	IIS configuration mode 00: Slave - transmit 01: Slave - receive 10: Master - transmit 11: Master - receive **NOTE:** This bit should be configured when the IIS is disabled. It is not used in SPI mode.
7	IISPCMSYNCSEL	PCM frame synchronization 0: Short frame synchronization 1: Long frame synchronization **NOTE:** This bit only has meaning if IISSTD = 11 (PCM standard is used). It is not used in SPI mode.
6	Reserved	Reserved
5:4	IISSTD	IIS standard selection 00: IIS Philips standard 01: MSB justified standard (left justified) 10: LSB justified standard (right justified) 11: PCM standard Not used in SPI mode. **NOTE:** These bits should be configured for correct operation when the IIS is disabled.
3	IISCKPL	Steady state clock polarity

Field	Name	Description
		0: IIS clock steady state is low level 1: IIS clock steady state is high level **NOTE:** This bit should be configured for correct operation when the IIS is disabled. This bit is not used in SPI mode.
2:1	IISDATLEN	Data length to be transferred 00: 16-bit data length 01: 24-bit data length 10: 32-bit data length 11: Not allowed **NOTE:** These bits should be configured for correct operation when the IIS is disabled. This bit is not used in SPI mode.
0	IISCHNLEN	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 is filled in. Not used in SPI mode. **NOTE:** This bit should be configured for correct operation when the IIS is disabled.

26.6.2.9 SPI_IISCFG1

0x0020			IIS configuration register 1											SPI_IISCFG1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						IISMCKOEN	IISODDSCALER	IISEVENSCALER							
Type	RO						RW	RW	RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 26-14 IIS Configuration Register 1 Description

Field	Name	Description
31:10	Reserved	Reserved
9	IISMCKOEN	Master clock output enable 0: Master clock output is disabled 1: Master clock output is enabled **NOTE:** This bit should be configured when the IIS is disabled. It is used only when the IIS is in master mode. This bit is not used in SPI mode.
8	IISODDSCALER	Odd factor for the prescaler 0: Real divider value is = IISDIV *2 1: Real divider value is = (IISDIV * 2)+1

Field	Name	Description
		Not used in SPI mode. **NOTE:** This bit should be configured when the IIS is disabled. It is used only when the IIS is in master mode.
7:0	IISEVENSCALER	IIS Linear prescaler IISDIV [7:0] = 0 or IISDIV [7:0] = 1 are forbidden values. Not used in SPI mode. **NOTE:** These bits should be configured when the IIS is disabled. It is used only when the IIS is in master mode.

26.6.2.10 SPI_EXTCFG0

0x0024			SPI extra configuration register 0											SPI_EXTCFG0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FIFOBUFSEL	Reserved	IISLRCKPOLAR	IISFDEN	IISHDEN	Reserved				IISDAT20BIT	RXCLKREVTS	IISCLKREVTM	IISLRCKRFIRST	IISLRCKSELEN	ENDIANSEL	Reserved
Type	RW	RO	RW	RW	RW	RO				RW	RW	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-15 SPI Extra Configuration Register 0 Description

Field	Name	Description
31:16	Reserved	Reserved
15	FIFOBUFSEL	FIFO buffer function selection This bit is set and cleared by software. 0: FIFO mode is disabled. 1: FIFO mode is enabled. **NOTE:** These bits should be configured for correct operation when the SPI and IIS are disabled.
14	Reserved	Reserved
13	IISLRCKPOLAR	lrck polarity 0: IIS lrck state is low level 1: IIS lrck state is high level

Field	Name	Description
		NOTE: This bit should be configured for correct operation when the IIS is disabled. This bit is not used in SPI mode.
12	IISFDEN	IIS full duplex mode enable 0: IIS full duplex mode disable 1: IIS full duplex mode enable Bypass IISMODE configuration
11	IISHDEN	IIS half duplex mode enable 0: IIS half duplex mode disable 1: IIS half duplex mode enable Configure IISMODE to select direction of transfer
10:7	Reserved	Reserved
6	IISDAT20BIT	20-Bit Data length to be transferred 0: Data length depend on IISDATLEN 1: 20-bit data length
5	RXCLKREVTS	Rx clock revert 0: SCK standard edge sampling 1: SCK standard edge is shifted back by one edge sampling (only supports all SPI non-TI modes, all IIS non-pcm modes)
4	IISCLKREVTM	IIS clock revert master 0: IIS output on SCK standard edge 1: IIS output on sck inversion edge (Only valid in Philips standard, LSB standard, MSB standard mode)
3	IISLRCKRFIRST	lrck output right channel first 0: lrck outputs the left channel first

Field	Name	Description
		1: lrck outputs the right channel first (Only valid in Philips standard, LSB standard, MSB standard mode)
2	IISLRCKSELEN	lrck selection enable 0: lrck output in standard mode 1: lrck output depend on IISLRCKRFIRST configuration
1	ENDIANSEL	Endian selection 0: Bus data little-endian selection 1: Bus data big-endian selection
0	Reserved	Reserved

26.6.2.11 SPI_FIFOCFG

0x0028			SPI FIFO configuration register											SPI_FIFOCFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					RXFAFLVL			Reserved	TXFAELVL			Reserved			
Type	RO					RW			RO	RW			RO			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-16 SPI FIFO Configuration Register Description

Field	Name	Description
31:11	Reserved	Reserved
10:8	RXFAFLVL	RXFIFO threshold configuration
7	Reserved	Reserved
6:4	TXFAELVL	TXFIFO threshold configuration
3:0	Reserved	Reserved

26.6.2.12 SPI_CRCCFG

0x002c			SPI CRC initial configuration register											SPI_CRCCFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CRCINIT															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-17 SPI CRC Initial Configuration Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CRCINIT	CRC polynomial register This register contains the initial value for the CRC calculation.

26.6.2.13 SPI_EXTCFG1

0x0030			SPI extra configuration register 1											SPI_EXTCFG1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		CRCXOROUT	LENCFGEN	LENCFG				DUMMYEN	DUMMYLEN			Reserved			
Type	RO		RW	RW	RW				RW	RW			RO			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-18 SPI Extra Configuration Register 1 Description

Field	Name	Description
31:14	Reserved	Reserved
13	CRCXOROUT	CRC output XOR 0: CRC output XOR 0x0 1: CRC output XOR 0xFFFF
12	LENCFGEN	SPI length cfg enable 0: SPI length depend on FRMFORMAT 1: Length depend on LENCFG
11:8	LENCFG	SPI length Data size These bits configure the data length for SPI transfers. 0000: Not used 0001: Not used 0010: Not used

Field	Name	Description
		0011: 4-bit 0100: 5-bit 0101: 6-bit 0110: 7-bit 0111: 8-bit 1000: 9-bit 1001: 10-bit 1010: 11-bit 1011: 12-bit 1100: 13-bit 1101: 14-bit 1110: 15-bit 1111: 16-bit **NOTE:** These bits are not used in IIS mode.
7	DUMMYEN	SPI CLK dummy enable 0: SPI CLK dummy disable 1: SPI CLK dummy enable
6:4	DUMMYLEN	SPI CLK dummy bit configuration
3:0	Reserved	Reserved

26.6.2.14 SPI_EXTCFG2

0x0034			SPI extra configuration register 2											SPI_EXTCFG2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	NSSSEL		Reserved						SCKDLYEN	SCKDLYLEN						
Type	RW		RO						RW	RW						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-19 SPI Extra Configuration Register 2 Description

Field	Name	Description
31:16	Reserved	Reserved
15:13	NSSSEL	NSS channel selection 00: Channel 0 01: Channel 1 10: Channel 2 11: Channel 3
12:8	Reserved	Reserved
7	SCKDLYEN	SPI master SCK delay enable 0: SPI master SCK delay disable 1: SPI master SCK delay enable
6:0	SCKDLYLEN	SPI master SCK delay configuration

26.6.2.15 SPI_EXTCFG3

0x0038			SPI extra configuration register 3											SPI_EXTCFG3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DATDLYEN	DATDLYLEN							NSSDLYEN	NSSDLYLEN						
Type	RW	RW							RW	RW						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26-20 SPI Extra Configuration Register 3 Description

Field	Name	Description
31:16	Reserved	Reserved
15	DATDLYEN	SPI master DAT delay enable 0: SPI master DAT delay disable 1: SPI master DAT delay enable
14:8	DATDLYLEN	SPI master DAT delay configuration
7	NSSDLYEN	SPI master NSS delay enable 0: SPI master NSS delay disable 1: SPI master NSS delay enable
6:0	NSSDLYLEN	SPI master NSS delay configuration

Voltage Reference Buffer (VREFBUF)

This chapter describes the details of the Voltage Reference Buffer (VREFBUF).

Topics: **Page**

27.1 Introduction

The Voltage Reference Buffer (VREFBUF) module acts as a voltage reference buffer, providing voltage references for ADCs, DACs, and CMPs. It also functions as a voltage reference for external components through the VREF+ pin. Meanwhile, the VREF+ pin can be used as external input voltage reference for internal modules and VREFBUF must be disabled.

27.2 Features

- Three output voltage options: 1.5 V, 2.0 V, and 2.5 V
- Bypass mode to use V_{DDA} as the reference voltage
- Supports both internal and external voltage references
- Maximum load current: Up to 5 mA

27.3 Functional Description

The VREFBUF module is configured by software and serves as a voltage reference for both internal modules and external components.

27.3.1 Block Diagram

Figure 27-1 shows the block diagram of the VREFBUF module.

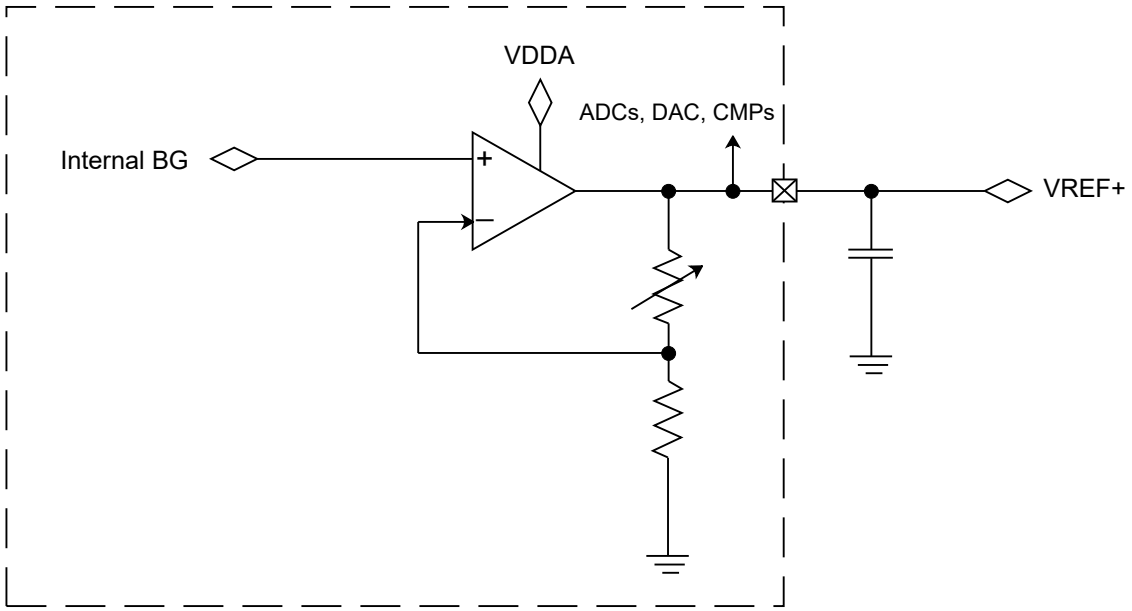


Figure 27-1 VREFBUF Block Diagram

27.3.2 Operation Mode

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below.

Table 27-1 lists the supported configurations for the VREFBUF.

Table 27-1 Supported Configurations

ENVR	HIZ	Description
0	0	VREFBUF buffer off mode:

ENVR	HIZ	Description
		• V_{REF+} pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): • VREFBUF buffer off • V_{REF+} pin input mode
1	0	Internal voltage reference mode: • VREFBUF buffer on • V_{REF+} pin connected to VREFBUF buffer output
1	1	Hold mode: • VREFBUF buffer off • V_{REF+} pin floating. The voltage is held with the external capacitor. • VRR detection is disabled, and the VRR bit keeps the last state.
VRBYP = 1		When ENVR = 1 and HIZ = 0, the VREFBUF buffer is off, VREFBUF output is V_{DDA}.

After enabling the VREFBUF by setting the ENVR bit and clearing the HIZ bit in the VREFBUF_CSR register, the user must wait until the VRR bit is set, indicating that the voltage reference output has reached its expected value.

27.3.3 VREFBUF Trimming

The VREFBUF output voltage is factory-calibrated by 3PEAK. After each reset and whenever the VRS setting is changed, the calibration value is automatically loaded to the TRIM register. The user can trim the output voltage by directly changing the TRIM register bits. In this case, the VRS setting will no longer affect the TRIM register until the device is reset.

27.4 Registers

27.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	VREFBUF_CSR	VREFBUF control and status register
0x0004	VREFBUF_CCR	VREFBUF calibration control register

27.4.2 Register Field Details

27.4.2.1 VREFBUF_CSR

0x0000			VREFBUF control and status register											VREFBUF_CSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved									VROCPPD	VRBYP	VRS		VRR	HIZ	ENVR
Type	RO									RW	RW	RW		RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 27-2 VREFBUF Control and Status Register Description

Field	Name	Description
31:7	Reserved	Reserved
6	VROCPPD	VREFBUF Overcurrent Protection (OCP) disable 0: OCP enable

Field	Name	Description
		1: OCP disable
5	VRBYP	VREFBUF bypass mode control 0: Disabled 1: Enabled, buffer off, VREFBUF output is VDDA **NOTE:** This bit is only valid when ENVR =1 and HIZ = 0.
4:3	VRS	Voltage reference scale These bits select the value generated by the VREFBUF. 00: Voltage reference set to 1.5 V 01: Voltage reference set to 2.0 V 10: Voltage reference set to 2.5 V 11: Reserved **NOTE:** The software can program this bitfield only when the VREFBUF is disabled (ENVR = 0).
2	VRR	VREFBUF ready 0: VREFBUF output is not ready. 1: VREFBUF output reached the requested level.
1	HIZ	High impedance mode This bit controls the analog switch that determines whether to connect the V_{REF+} pin or not. 0: V_{REF+} pin is internally connected to the VREFBUF output. 1: V_{REF+} pin is high impedance. VREFBUF modes for the mode descriptions depending on ENVR bit configuration.
0	ENVR	VREFBUF mode enable This bit is used to enable the VREFBUF mode. 0: Internal voltage reference mode disable (external voltage reference mode).

Field	Name	Description
		1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

27.4.2.2 VREFBUF_CCR

0x0004			VREFBUF calibration control register											VREFBUF_CCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										TRIM					
Type	RO										RW					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27-3 VREFBUF Calibration Control Register Description

Field	Name	Description
31:6	Reserved	Reserved
5:0	TRIM	Trimming code The TRIM code is a 6-bit data that is set and updated according to the mechanism described below. Reset: TRIM[5:0] is automatically initialized with the VRS = 0 trimming value stored in the Flash memory during the production test. VRS change:

Field	Name	Description
		TRIM[5:0] is automatically initialized with the trimming value (corresponding to the VRS setting) stored in the Flash memory during the production test. Write in TRIM[5:0]: User can modify the TRIM[5:0] with an arbitrary value. This permanently disables the trimming value's control with VRS until the device is reset.

TPSensor®

This chapter introduces the TPSensor® module. TPSensor® is a capacitive sensing technology developed by 3PEAK. The TPSensor® module is fully supported by the TPSensor® touch sensing firmware library.

The following declaration and terms are used to describe parts of the TPSensor® module:

Core	Capacitance to digital converter.
Channel	A channel is the physical connection between an external electrode and the corresponding TPSensor input switch.
CAPCHx	TPSENSOR channel (x = 0 to 15 in TPS325M51xx)
Rx	TPSensor channel configured as receive input (x = 0 to 15 in TPS325M51xx)
Tx	TPSensor channel configured as transmit output (x = 8 to 15 in TPS325M51xx)
Cp	Parasitic capacitance
Cs	In self-capacitance mode, this is the capacitor of unknown value refer to ground (which is the electrode or touchpad).
Cm	In mutual-capacitance mode, this is the capacitance between the Tx and Rx electrodes.
Cx	Represents Cs or Cm
Raw count	The digital output value of the capacitance to the digital converter. The value is proportional to the capacitance measured.
Conversion	Capacitance measurement by TPSensor. From a hardware perspective, the conversion translates the capacitance to raw count.

Topics: **Page**

28.1 Introduction

TPSensor is a touch sensing peripheral that is attached to a 3PEAK microcontroller.

The TPSensor module supports relative capacitance measurement by detecting capacitance changes.

28.2 Features

- Charge-transfer capacitance measurement technology
- Self-capacitance and Mutual-capacitance measurement
- Supports up to 16 capacitive sensing channels
- Dedicated shield channel and guard channel
- Compensation for parasitic capacitance
- Support up to 200 pF parasitic capacitance
- Up to 10-bit resolution with OSR mode
- Integrated calibration capacitors
- Frequency hopping provides common mode noise immunity
- Active mode and sleep mode
- Synchronized conversion (external trigger)
- Wake on proximity: Wakes the system from a low-power mode to active mode on a proximity event
- Designed to operate with TPSensor touch sensing firmware library

28.3 Functional Description

28.3.1 Block Diagram

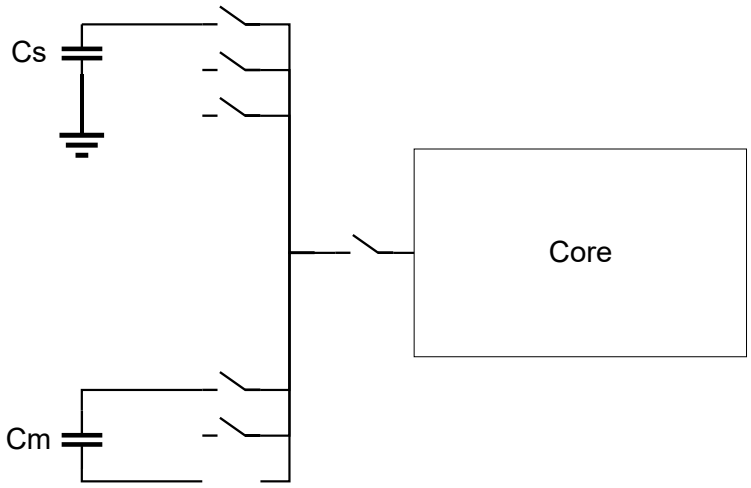


Figure 28-1 TPSensor Block Diagram

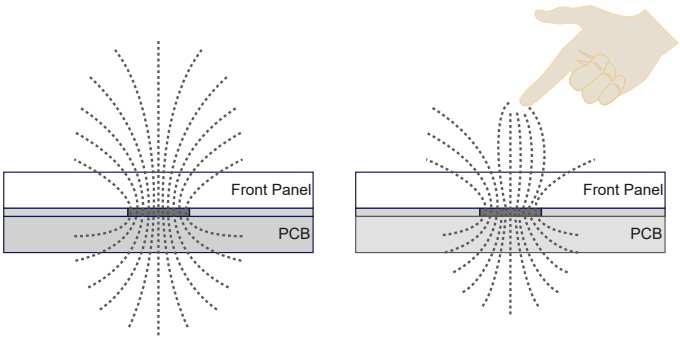
28.3.2 Operation Modes

28.3.2.1 Self-Capacitance Mode

Self-capacitance refers to a capacitive measurement using a single sensor electrode to measure the apparent capacitance between the electrode and the DC ground of the touch sensor MCU circuit.

The base capacitance is formed by the combination of parasitic, sensor and ground return capacitance. In combination, these forms the “No touch” or default capacitance that is measured during calibration and is used as a reference to detect a capacitance change indicating touch contact.

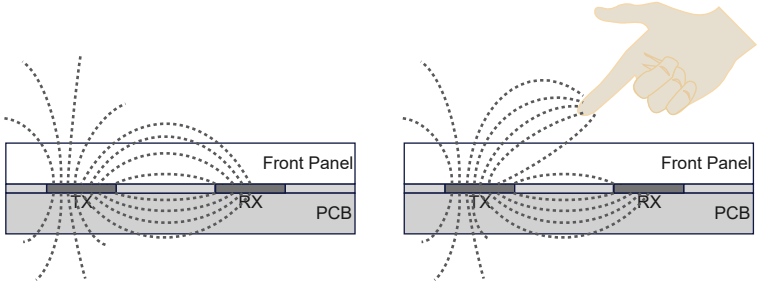
When a touch contact is applied, the apparent sensor capacitance is increased by the introduction of a parallel path to earth through the human body. This increase is referred to as the touch “Delta”.



28.3.2.2 Mutual-Capacitance Mode

Mutual capacitance refers to a capacitive measurement using a pair of sensor electrode to measure the apparent capacitance between them. Typically, one electrode acts as the Driver (Tx), while the other is the Receiver (Rx). The sensor is formed where the electrodes are placed close together.

When a touch contact is applied to the area on the panel where a Tx meets an Rx, the mutual capacitance between those Tx and Rx electrodes is reduced. This is because the user’s interaction has the effect of disturbing the electric field propagation between the two electrodes.



28.3.3 Operation Mode Configuration

All the operation modes are controlled by configuration register TPSENSOR_CORE_CCFR. There are 18 channels connected to TPSensor core.

- 8x Rx channel
- 8x Rx/Tx channel
- 1x guard channel
- 1x shield channel

Each Rx channel can work individually or bond together. Rx channels bonding feature is controlled by BONDEN bit in control register TPSENSOR_TCR and RXMUX bits in register TPSENSOR_TRXR. Tx channels are only used in mutual-mode measurement.

The guard channel is usually connected to the guard sensor. A guard sensor is a copper trace that surrounds all the sensors on the PCB. The guard sensor is used to detect large objects as well as spills. The guard channel works in self-mode. Theoretically, any Rx channel can work as a guard channel.

The shield channel is usually connected to the shield electrode. A shield electrode is a hatched fill that is driven with a signal which is the replica of the sensor signal to reduce the parasitic capacitance of sensors for liquid tolerance and for proximity sensing.

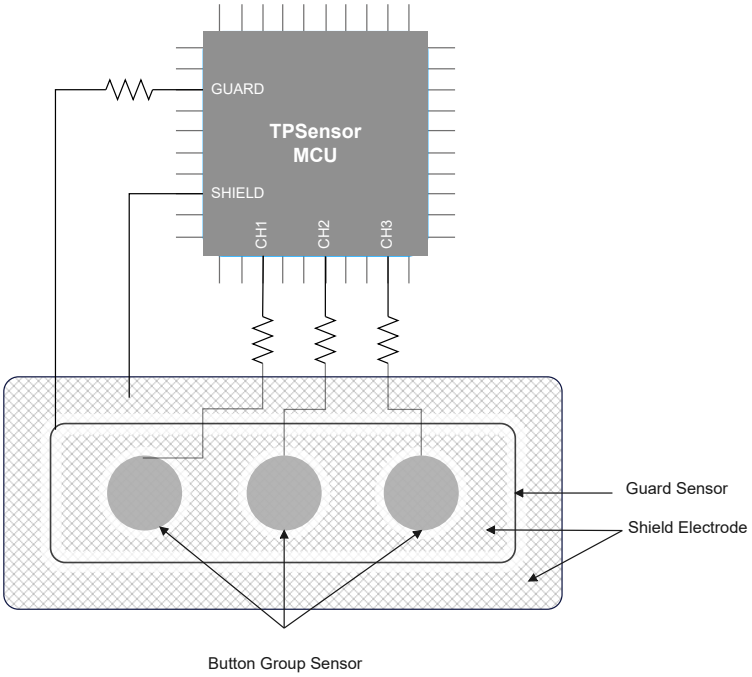


Figure 28-2 Guard Sensor and Shield Electrode

Channel operation mode is configurable by MODE bits in register TPSENSOR_CORE_CCFR. These configuration modes are only available for corresponding channel. For example, the shield mode is only available for shield channel.

28.3.4 Conversion

28.3.4.1 Single Conversion

The conversion result (raw count) is saved in register TPSENSOR_CDR.

28.3.4.2 OSR Conversion

TPSensor supports OSR (over sampling) mode to improve measuring quantization error in single conversion measurement thus increasing the measurement resolution. In OSR mode, TPSensor runs OSR times conversion automatically with single trigger and outputs accumulated raw count in register TPSENSOR_CDR.

OSR is configurable by FUNCOSC bits in register TPSENSOR_TANAR.

28.3.5 Conversion Trigger

28.3.5.1 Software Trigger

Conversion is automatically triggered when the TPSENSOREN bit in TPSENSOR_TCR is set, and the conversion control register (TPSENSOR_CORE_CCFR) is properly configured.

28.3.5.2 External Trigger

To initiate conversions synchronized with an external event, use the external trigger feature. Activate the synchronization capability by setting the EXTTRIGEN bit in TPSENSOR_TCR to 1. Once you

set the TPSENSOREN bit in TPSENSOR_TCR to 1 and configure the TPSENSOR_CORE_CCFR correctly, the conversion will commence instantly upon detecting a valid edge on the SYNC synchronization input.

28.3.6 Low-Power Mode Operation

TPSensor supports low-power mode operation, which allows for periodic measurement and analysis of a touch element without CPU intervention. If the analysis determines an event requiring further processing or decision-making, an interrupt is generated to wake up the CPU.

In sleep mode, the TPSensor supports single or bonded Rx channels. The enable of channels is controlled via RXMUX bits in the TPSENSOR_TRXR register. However, TPSensor does not support scanning across multiple channels while in sleep mode.

28.3.6.1 Clock

In low-power mode, IHS from the digital portion is turned off, and a 10 MHz OSC from the analog portion provides the clock for the TPSensor module.

28.3.6.2 Threshold

In low power mode, the threshold is set by SLEEPTH bits in register TPSENSOR_SLPCFG. SLEEPHYST bits in register TPSENSOR_SLPHYST provide immunity against noisy transitions of detected state.

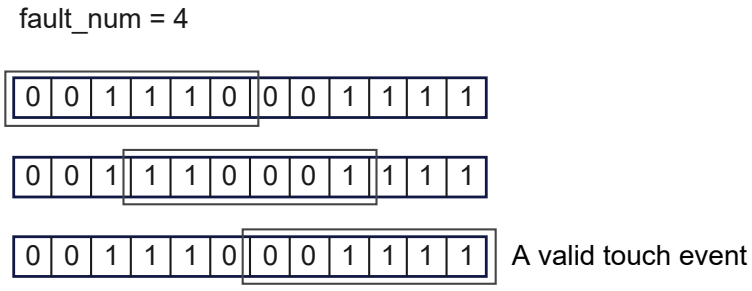
$$\text{detected state} = \text{ON} \quad \text{if}(\text{raw count} \geq \text{SLEEP\_TH} + \text{SLEEP\_HYST}) \tag{15}$$

$$\text{detected state} = \text{OFF} \quad \text{if}(\text{raw count} \leq \text{SLEEP\_TH} - \text{SLEEP\_HYST}) \tag{16}$$

28.3.6.3 Sliding Window Process

To ensure robust operation in noisy environments, TPSensor provides a sliding window process for touch detection. It works by requiring a sensor going through a state change (no-touch into touch) to be in the new state for a certain number of samples within a given window size before reporting a valid touch event.

Take the plot below as an example. The measurement results (over time for several samples) are plotted in the state series. 0 represents the OFF state and 1 represents the ON state. The blue box is the sliding window, the window size is 6. fault_num determines how many ON state samples are required to report a valid touch event within the sliding window.



Sliding window size is controlled by SLEEPWIN bits in register TPSENSOR_SLPCFG. Fault_num is controlled by FAULTNUM bits in register TPSENSOR_SLPCFG.

28.3.6.4 Scan Rate

In sleep mode, a periodic conversion is triggered by a low-frequency clock, which can be selected from either an internal 32 kHz clock (ILS) or an external 32 kHz crystal (ELS).

LSCLKEN	LSCLKSEL	32 kHz Clock Source
0	X	NA
1	0	Internal 32 kHz Clock
1	1	Crystal

The scan interval in low power mode is set by SLEEPSCANINTERV bits in register TPSENSOR_SLPCFG. To convert to Samples per Second (SPS), simply take 1000 divided by the specified interval. For example, an interval (response time) of 50 milliseconds equals 20 SPS.

SLEEPSCANINTERV[14:12]	000	001	010	011	100	101	110	111
Scan Interval (ms)	0	20	40	60	80	100	120	200

28.4 Interrupts

28.4.1 Overview

Two interrupt registers are associated with the TPSensor module. One for active mode and the other for sleep mode.

All interrupt flags request an interrupt when their corresponding interrupt enable bit and the global interrupt enable bit are set.

28.4.2 Active Mode Interrupt

Enable Bit	Interrupt Flag	Description
ACTIVEINTEN	ACTIVEINT	End of conversion flag
INTERFINTEN	INTERFINT	Noise interrupt flag
CFGFIFOINTEN	CFGFIFOOVF	Configuration FIFO overflow flag
	CFGFIFOUDF	Configuration FIFO underflow flag
DATAFIFOINTEN	DATAFIFOOVF	Data FIFO overlay flag
	DATAFIFOUDF	Data FIFO underflow flag

28.4.3 Sleep Mode Interrupt

Enable Bit	Interrupt Flag	Description
SLEEPINTEN	SLEEPINT	Sleep mode detection interrupt flag

28.4.4 Clearing Interrupt Flag

All the interrupt flags need to be clear manually. Write a 1b to the interrupt flag register will clear interrupt flag.

28.5 Registers

28.5.1 Register Address Map

Offset	Register Name	Register Description
0x0000	TPSENSOR_TISR	Interrupt register
0x0004	TPSENSOR_TIER	Interrupt enable
0x0008	TPSENSOR_TCR	Control register
0x000c	TPSENSOR_TCKR	Clock control register
0x0010	TPSENSOR_TRXR	RX control register
0x0014	TPSENSOR_TDLYR	Delay time control register
0x0018	TPSENSOR_TSIVR	Scan interval control register
0x001c	TPSENSOR_TANAR	Analog control register
0x0020	TPSENSOR_SLPCFG	Sleep control register
0x0024	TPSENSOR_SLPTH	Sleep threshold register
0x0028	TPSENSOR_SLPHYST	Sleep hysteresis register
0x002c	TPSENSOR_COMP0	Compensation time register
0x0030	TPSENSOR_COMP1	Compensation time register
0x0034	TPSENSOR_COMP2	Compensation time register
0x0038	TPSENSOR_COMP3	Compensation time register
0x003c	TPSENSOR_COMP4	Compensation time register
0x0040	TPSENSOR_COMP5	Compensation time register
0x0044	TPSENSOR_COMP6	Compensation time register

Offset	Register Name	Register Description
0x0048	TPSENSOR_COMP7	Compensation time register
0x004c	TPSENSOR_COMP8	Compensation time register
0x0050	TPSENSOR_TRIM	Sleep current calibration register
0x0054	TPSENSOR_TSTAT	Status register
0x0058	TPSENSOR_TDR	Data result
0x005c	TPSENSOR_TDR0	Sleep history result
0x0060	TPSENSOR_TDR1	Sleep history result
0x0064	TPSENSOR_TDR2	Sleep history result
0x0068	TPSENSOR_TDR3	Sleep history result
0x006c	TPSENSOR_TDR4	Sleep history result
0x0070	TPSENSOR_TDR5	Sleep history result
0x0074	TPSENSOR_TDR6	Sleep history result
0x0078	TPSENSOR_TDR7	Sleep history result
0x007c	TPSENSOR_TDR8	Sleep history result
0x0080	TPSENSOR_TDR9	Sleep history result
0x0084	TPSENSOR_TDR10	Sleep history result
0x0088	TPSENSOR_TDR11	Sleep history result
0x008c	TPSENSOR_TDR12	Sleep history result
0x0090	TPSENSOR_TDR13	Sleep history result
0x0094	TPSENSOR_TDR14	Sleep history result
0x0098	TPSENSOR_TDR15	Sleep history result

Offset	Register Name	Register Description
0x009c	TPSENSOR_TPD0	Pad ground register
0x00a0	TPSENSOR_TPD1	Pad ground register
0x00a4	TPSENSOR_TSST	Sleep mode setup time

Offset	Register Name	Register Description
0x0100	TPSENSOR_CORE_CISR	Interrupt register
0x0104	TPSENSOR_CORE_CIER	Interrupt enable register
0x0108	TPSENSOR_CORE_CCR	Data FIFO control register
0x010c	TPSENSOR_CORE_CSTAT	FIFO status register
0x0110	TPSENSOR_CORE_CCTR0	Capacitance calibration result register
0x0114	TPSENSOR_CORE_CCTR1	Capacitance calibration result register
0x0118	TPSENSOR_CORE_CCTR2	Capacitance calibration result register
0x011c	TPSENSOR_CORE_CCTR3	Capacitance calibration result register
0x0180	TPSENSOR_CORE_CCFR	CFG FIFO register
0x0184	TPSENSOR_CORE_CDR	Data FIFO register

28.5.2 Register Field Details

28.5.2.1 TPSENSOR_TISR

0x0000			Interrupt register											TPSENSOR_TISR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															SLEEPINT
Type	RO															RC_W1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-1 Interrupt Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	SLEEPINT	Sleep mode interrupt

28.5.2.2 TPSENSOR_TIER

0x0004			Interrupt enable register											TPSENSOR_TIER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved															SLEEPINTEN
Type	RO															RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-2 Interrupt Enable Register Description

Field	Name	Description
31:1	Reserved	Reserved
0	SLEEPINTEN	Sleep mode interrupt enable 0: Disable 1: Enable

28.5.2.3 TPSENSOR_TCR

0x0008			Control register											TPSENSOR_TCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	CKVDDEN	Reserved	SLEEPMODELP	PAGUARDCOMPEN	FUNCOSR					SLEEPMODE	SLEEPEN	TPSENSOREN	BONDEN	EXTTRIGEN	Reserved
Type	RO	RW	RO	RW	RW	RW					RW	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-3 Control Register Description

Field	Name	Description
31:15	Reserved	Reserved
14	CKVDDEN	Analog switch control (CK_COMP_VDD_VDD)
13	Reserved	Reserved
12	SLEEPMODELP	Sleep mode without VREFBUF
11	PAGUARDCOMPEN	Compensation enable 1: Enable
10:6	FUNCOSR	Over sample rate 0000: 1 0001: 2 0010: 4

Field	Name	Description
		0011: 8 0100: 16 0101: 32 0110: 64
5	SLEEPMODE	0: Without compensation 1: With compensation
4	SLEEPEN	Sleep mode enable 0: Disable 1: Enable The register related to sleep mode cannot be configured when SLEEPEN = 1.
3	TPSENSOREN	TPSensor enable 0: Disable 1: Enable
2	BONDEN	RX pad bonding enable 0: Unbonding 1: Boding
1	EXTTRIGEN	External trigger enable 0: Disable 1: Enable
0	Reserved	Reserved

28.5.2.4 TPSENSOR_TCKR

0x000c			Clock control register											TPSENSOR_TCKR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														LSCLKEN	LSCLKSEL
Type	RO														RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-4 Clock Control Register Description

Field	Name	Description
31:2	Reserved	Reserved
1	LSCLKEN	32K clock enable 0: Disable 1: Enable
0	LSCLKSEL	ELS source selection 0: ILS 1: ELS

28.5.2.5 TPSENSOR_TRXR

0x0010			RX control register											TPSENSOR_TRXR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RXMUX															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-5 RX Control Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	RXMUX	This register is used to select RX pad in bonding mode; This register is used to select RX pad in sleep mode.

28.5.2.6 TPSENSOR_TDLYR

0x0014			Delay time control register											TPSENSOR_TDLYR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RAWCNTEN	Reserved	TRANSTIME							Reserved	CHARGETIME					
Type	RW	RO	RW							RO	RW					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-6 Delay Time Control Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	RAWCNTEN	Over sampling data accumulation enable 0: Enable, accumulated 1: Disable, not accumulated
14	Reserved	Reserved
13:7	TRANSTIME	Charge transfer time transfer_time = (TRANSTIME +1) * 100ns
6	Reserved	Reserved
5:0	CHARGETIME	Capacitance charge time charge_time = (CHARGETIME +1) * 100ns

28.5.2.7 TPSENSOR_TSIVR

0x0018			Scan interval control register											TPSENSOR_TSIVR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	CHSCANINTERVOFFSET			CHSCANINTERVLSH		Reserved	CHSCANINTERV								
Type	RO	RW			RW		RO	RW								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-7 Scan Interval Control Register Description

Field	Name	Description
31:15	Reserved	Reserved
14:12	CHSCANINTERVOFFSET	Scan interval offset in over sampling mode Offset = CHSCANINTERVOFFSET * 100ns
11:10	CHSCANINTERVLSH	Scan interval base num left shift bits 00: No shifter 01: Left shifter 1-bit 10: Left shifter 2-bit 11: Left shifter 3-bit
9	Reserved	Reserved
8:0	CHSCANINTERV	Scan interval base num, (CHSCANINTERV +1)*100ns

28.5.2.8 TPSENSOR_TANAR

0x001c			Analog control register											TPSENSOR_TANAR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		ANALOGDITHER		Reserved	Reserved				IDAC2SEL			Reserved	IDAC1SEL		
Type	RO		RW		RO	RO				RW			RO	RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-8 Analog Control Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:12	ANALOGDITHER	Analog scan interval offset 01: 0ns 01: 10ns 10: 20ns 11: 30ns
11	Reserved	Reserved
10:7	Reserved	Reserved
6:4	IDAC2SEL	Idac2 current selection 0000: 0 uA 0001: 1 uA

Field	Name	Description
		0010: 2 uA 0011: 4 uA 0100: 8 uA 0101: 20 uA 0110: 50 uA
3	Reserved	Reserved
2:0	IDAC1SEL	Idac1 current selection 0000: 0 uA 0001: 1 uA 0010: 2 uA 0011: 4 uA 0100: 8 uA 0101: 20 uA 0110: 50 uA

28.5.2.9 TPSENSOR_SLPCFG

0x0020			Sleep control register											TPSENSOR_SLPCFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved	SLEEPSCANINTERV			Reserved		FAULTNUM					SLEEPWIN				
Type	RO	RW			RO		RW					RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-9 Sleep Control Register Description

Field	Name	Description
31:15	Reserved	Reserved
14:12	SLEEPSCANINTERV	Sleep mode scan interval 000: 0ms 001: 20ms 010: 40ms 011: 60ms 100: 80ms 101: 100ms 110: 120ms 111: 200ms
11:10	Reserved	Reserved

Field	Name	Description
9:5	FAULTNUM	The threshold of sliding window in sleep mode
4:0	SLEEPWIN	The width of sliding window in sleep mode

28.5.2.10 TPSENSOR_SLPTH

0x0024			Sleep threshold register											TPSENSOR_SLPTH		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		SLEEPTH													
Type	RO		RW													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-10 Sleep Threshold Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	SLEEPTH	Threshold in sleep mode

28.5.2.11 TPSENSOR_SLPHYST

0x0028			Sleep hysteresis register											TPSENSOR_SLPHYST		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		SLEEPHYST													
Type	RO		RW													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-11 Sleep Hysteresis Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	SLEEPHYST	Threshold hysteresis in sleep mode

28.5.2.12 TPSENSOR_COMP0

0x002c			Compensation time register											TPSENSOR_COMP0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COMPNUM1								COMPNUM0							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-12 Compensation Time Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	COMPNUM1	RX pad1 compensation number, COMPNUM1 * 600ns
7:0	COMPNUM0	RX pad0 compensation number, COMPNUM0 * 600ns

28.5.2.13 TPSENSOR_COMP1

0x0030			Compensation time register											TPSENSOR_COMP1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COMPNUM3								COMPNUM2							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-13 Compensation Time Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	COMPNUM3	RX pad3 compensation number, COMPNUM3 * 600ns
7:0	COMPNUM2	RX pad2 compensation number, COMPNUM2 * 600ns

28.5.2.14 TPSENSOR_COMP2

0x0034			Compensation time register											TPSENSOR_COMP2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COMPNUM5								COMPNUM4							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-14 Compensation Time Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	COMPNUM5	RX pad5 compensation number, COMPNUM5 * 600ns
7:0	COMPNUM4	RX pad4 compensation number, COMPNUM4 * 600ns

28.5.2.15 TPSENSOR_COMP3

0x0038			Compensation time register											TPSENSOR_COMP3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COMPNUM7								COMPNUM6							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-15 Compensation Time Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	COMPNUM7	RX pad7 compensation number, COMPNUM7 * 600ns
7:0	COMPNUM6	RX pad6 compensation number, COMPNUM6 * 600ns

28.5.2.16 TPSENSOR_COMP4

0x003c			Compensation time register											TPSENSOR_COMP4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COMPNUM9								COMPNUM8							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-16 Compensation Time Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	COMPNUM9	RX pad9 compensation number, COMPNUM9 * 600ns
7:0	COMPNUM8	RX pad8 compensation number, COMPNUM8 * 600ns

28.5.2.17 TPSENSOR_COMP5

0x0040			Compensation time register											TPSENSOR_COMP5		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COMPNUM11								COMPNUM10							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-17 Compensation Time Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	COMPNUM11	RX pad11 compensation number, COMPNUM11 * 600ns
7:0	COMPNUM10	RX pad10 compensation number, COMPNUM10 * 600ns

28.5.2.18 TPSENSOR_COMP6

0x0044			Compensation time register											TPSENSOR_COMP6		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COMPNUM13								COMPNUM12							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-18 Compensation Time Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	COMPNUM13	RX pad13 compensation number, COMPNUM13 * 600ns
7:0	COMPNUM12	RX pad12 compensation number, COMPNUM12 * 600ns

28.5.2.19 TPSENSOR_COMP7

0x0048			Compensation time register											TPSENSOR_COMP7		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COMPNUM15								COMPNUM14							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-19 Compensation Time Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	COMPNUM15	RX pad15 compensation number, COMPNUM15 * 600ns
7:0	COMPNUM14	RX pad14 compensation number, COMPNUM14 * 600ns

28.5.2.20 TPSENSOR_COMP8

0x004c			Compensation time register											TPSENSOR_COMP8		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	GUARDCOMPNUM								BONDCOMPNUM							
Type	RW								RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-20 Compensation Time Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:8	GUARDCOMPNUM	Guard bonded pads compensation number, GUARDCOMPNUM * 600ns
7:0	BONDCOMPNUM	RX bonded pads compensation number, BONDCOMPNUM * 600ns

28.5.2.21 TPSENSOR_TRIM

0x0050			Sleep current calibration register											TPSENSOR_TRIM		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved						SLEEP1UATRIM									
Type	RO						RW									
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0

Table 28-21 Interrupt Enable register Description

Field	Name	Description
31:10	Reserved	Reserved
9:0	SLEEP1UATRIM	Current trim data in sleep mode

28.5.2.22 TPSENSOR_TSTAT

0x0054			Status register											TPSENSOR_TSTAT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	VDDFLAG	Reserved										ADDROFFSET				
Type	RO	RO										RO				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-22 Status Register Description

Field	Name	Description
31:16	Reserved	Reserved
15	VDDFLAG	Analog VDD flag
14:5	Reserved	Reserved
4:0	ADDROFFSET	Sleep result data address offset(0~15); address of data result0 is base address

28.5.2.23 TPSENSOR_TDR

0x0058			Data result											TPSENSOR_TDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-23 Data Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT	Sleep mode latest counter result

28.5.2.24 TPSENSOR_TDR0

0x005c			Sleep history result											TPSENSOR_TDR0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT0													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-24 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT0	History result in sleep mode

28.5.2.25 TPSENSOR_TDR1

0x0060			Sleep history result											TPSENSOR_TDR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT1													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-25 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT1	History result in sleep mode

28.5.2.26 TPSENSOR_TDR2

0x0064			Sleep history result											TPSENSOR_TDR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT2													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-26 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT2	History result in sleep mode

28.5.2.27 TPSENSOR_TDR3

0x0068			Sleep history result											TPSENSOR_TDR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT3													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-27 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT3	History result in sleep mode

28.5.2.28 TPSENSOR_TDR4

0x006c			Sleep history result											TPSENSOR_TDR4		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT4													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-28 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT4	History result in sleep mode

28.5.2.29 TPSENSOR_TDR5

0x0070			Sleep history result											TPSENSOR_TDR5		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT5													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-29 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT5	History result in sleep mode

28.5.2.30 TPSENSOR_TDR6

0x0074			Sleep history result											TPSENSOR_TDR6		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT6													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-30 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT6	History result in sleep mode

28.5.2.31 TPSENSOR_TDR7

0x0078			Sleep history result											TPSENSOR_TDR7		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT7													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-31 Sleep history result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT7	History result in sleep mode

28.5.2.32 TPSENSOR_TDR8

0x007c			Sleep history result											TPSENSOR_TDR8		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT8													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-32 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	SLEEPTH	Threshold in sleep mode

28.5.2.33 TPSENSOR_TDR9

0x0080			Sleep history result											TPSENSOR_TDR9		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT9													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-33 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT9	History result in sleep mode

28.5.2.34 TPSENSOR_TDR10

0x0084			Sleep history result											TPSENSOR_TDR10		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT10													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-34 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT10	History result in sleep mode

28.5.2.35 TPSENSOR_TDR11

0x0088			Sleep history result											TPSENSOR_TDR11		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT11													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-35 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT11	History result in sleep mode

28.5.2.36 TPSENSOR_TDR12

0x008c			Sleep history result											TPSENSOR_TDR12		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT12													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-36 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT12	History result in sleep mode

28.5.2.37 TPSENSOR_TDR13

0x0090			Sleep history result											TPSENSOR_TDR13		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT13													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-37 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT13	History result in sleep mode

28.5.2.38 TPSENSOR_TDR14

0x0094			Sleep history result											TPSENSOR_TDR14		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT14													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-38 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT14	History result in sleep mode

28.5.2.39 TPSENSOR_TDR15

0x0098			Sleep history result											TPSENSOR_TDR15		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		DATARESULT15													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-39 Sleep History Result Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	DATARESULT15	History result in sleep mode

28.5.2.40 TPSENSOR_TPD0

0x009c			Pad ground register											TPSENSOR_TPD0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RXPADGRDEN															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-40 Pad Ground Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	RXPADGRDEN	RX pad grounding enable

28.5.2.41 TPSENSOR_TPD1

0x00a0			Pad ground register											TPSENSOR_TPD1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														SHIELD PADGR DEN	GUARD PADGR DEN
Type	RO														RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-41 Pad Ground Register Description

Field	Name	Description
31:2	Reserved	Reserved
1	SHIELDPADGRDEN	Shield pad grounding enable
0	GUARDPADGRDEN	Guard pad grounding enable

28.5.2.42 TPSENSOR_TSST

0x00a4			Sleep mode setup time											TPSENSOR_TSST		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												SLEEPSETUPTIME			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-42 Sleep Mode Setup Time Register Description

Field	Name	Description
31:4	Reserved	Reserved
3:0	SLEEPSETUPTIME	Sleep mode setup time (SLEEPSETUPTIME * 2 + 3) 32K clk cycles

28.5.2.43 TPSENSOR_CORE_CISR

0x0100			Interrupt register											TPSENSOR_CORE_CISR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved										DATAFIFOUDF	DATAFIFOOVF	CFGFIFOUDF	CFGFIFOOVF	INTERFINT	ACTIVEINT
Type	RO										RC_W1	RC_W1	RC_W1	RC_W1	RC_W1	RC_W1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-43 Interrupt Register Description

Field	Name	Description
31:6	Reserved	Reserved
5	DATAFIFOUDF	Data FIFO over flow interrupt
4	DATAFIFOOVF	Data FIFO under flow interrupt
3	CFGFIFOUDF	Command FIFO over flow interrupt
2	CFGFIFOOVF	Command FIFO under flow interrupt
1	INTERFINT	Interference interrupt
0	ACTIVEINT	Actve counter intrrupt

28.5.2.44 TPSENSOR_CORE_CIER

0x0104			Interrupt enable register											TPSENSOR_CORE_CIER		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												DATAFIFOINTEN	CFGFIFOINTEN	INTERFINTEN	ACTIVEINTEN
Type	RO												RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-44 Interrupt Enable Register Description

Field	Name	Description
31:4	Reserved	Reserved
3	DATAFIFOINTEN	Data FIFO interrupt enable 0: Disable 1: Enable
2	CFGFIFOINTEN	Command FIFO interrupt enable 0: Disable 1: Enable
1	INTERFINTEN	Interference interrupt enable 0: Disable 1: Enable
0	ACTIVEINTEN	Actve counter intrrupt enable

Field	Name	Description
		0: Disable 1: Enable

28.5.2.45 TPSENSOR_CORE_CCR

0x0108			Data FIFO control register											TPSENSOR_CORE_CCR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												DATAFIFOTH			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-45 Data FIFO Control Register Description

Field	Name	Description
31:4	Reserved	Reserved
3:0	DATAFIFOTH	Data FIFO depth threshold

28.5.2.46 TPSENSOR_CORE_CSTAT

0x010c			FIFO status register											TPSENSOR_CORE_CSTAT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			DATAFIFODEPTH					Reserved			CFGFIFODEPTH				
Type	RO			RO					RO			RO				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-46 FIFO Status Register Description

Field	Name	Description
31:13	Reserved	Reserved
12:8	DATAFIFODEPTH	Data FIFO depth in active mode
7:5	Reserved	Reserved
4:0	CFGFIFODEPTH	Command FIFO depth in active mode

28.5.2.47 TPSENSOR_CORE_CCTR0

0x0110			Capacitance calibration result register											TPSENSOR_CORE_CCTR0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CAPCALITRIM0															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-47 Capacitance Calibration Result Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CAPCALITRIM0	Capacitance calibration result

28.5.2.48 TPSENSOR_CORE_CCTR1

0x0114			Capacitance calibration result register											TPSENSOR_CORE_CCTR1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CAPCALITRIM1															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-48 Capacitance Calibration Result Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CAPCALITRIM1	Capacitance calibration result

28.5.2.49 TPSENSOR_CORE_CCTR2

0x0118			Capacitance calibration result register											TPSENSOR_CORE_CCTR2		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CAPCALITRIM2															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-49 Capacitance Calibration Result Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CAPCALITRIM2	Capacitance calibration result

28.5.2.50 TPSENSOR_CORE_CCTR3

0x011c			Capacitance calibration result register											TPSENSOR_CORE_CCTR3		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CAPCALITRIM3															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-50 Capacitance Calibration Result Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	CAPCALITRIM3	Capacitance calibration result

28.5.2.51 TPSENSOR_CORE_CCFR

0x0180			CFG FIFO register											TPSENSOR_CORE_CCFR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved				RXSEL				Reserved	TXSEL			ACTIVEMODE			
Type	RO				RW				RO	RW			RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-51 CFG FIFO Register Description

Field	Name	Description
31:12	Reserved	Reserved
11:8	RXSEL	RX pad selection in unbonding mode 0000: RX1 0001: RX2 0010: RX3 0011: RX4 0100: RX5 0101: RX6 0110: RX7 0111: RX8 1000: RX9 1001: RX10

Field	Name	Description
		1010: RX11 1011: RX12 1100: RX13 1101: RX14 1110: RX15 1111: RX16
7	Reserved	Reserved
6:4	TXSEL	TX pad select 000: TX1 001: TX2 010: TX3 011: TX4 100: TX5 101: TX6 110: TX7 111: TX8
3:0	ACTIVEMODE	Active mode 0000: Self 0001: Shield 0010: Guard 0011: Mutual 0100: Interf signal 0101: Interf_noise 0110: Cali_step1 0111: Cali_step2 1000: Cali_step3 1001: Cali_step4

Field	Name	Description
		1010: Cali_step5 1011: Cali_step6 1100: Sleep_cali_step2

28.5.2.52 TPSENSOR_CORE_CDR

0x0184			Data FIFO register											TPSENSOR_CORE_CDR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved		ACTIVECNT													
Type	RO		RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 28-52 Data FIFO Register Description

Field	Name	Description
31:14	Reserved	Reserved
13:0	ACTIVECNT	Counter result in active mode

Controller Area Network (CAN)

This chapter describes the details of the Controller Area Network (CAN).

Topics: **Page**

29.1 Introduction

The Controller Area Network (CAN) bus is serial communication according to the CAN protocol and meets all constraints of its specification. This module supports CAN Classic (CAN 2.0A, CAN 2.0B) protocol.

29.2 Features

- Supports CAN Classic (2.0A, 2.0B), up to 8 bytes payload.
- Data rates up to 1 Mbit/s.
- Programmable baud rate prescaler (1 to 1/32).
- Separate clock domains for host interface, CAN protocol machine and time-stamping.
- Configurable receive buffer (RB) size:
  - Support up to 8 SLOTs.
  - FIFO-like behavior.
  - Received frames which are “not accepted” or “incorrect” do not overwrite already stored frames.
- Two Transmit Buffers:
  - Primary Transmit Buffer (PTB) (one frame slot).
  - Configurable Secondary Transmit Buffer (STB).
    - Support up to 8 SLOTs.
    - Operation in FIFO or priority decision mode.
- Independent and programmable internal acceptance filters:
  - Support up to 4 of acceptance filters.
- Extended features:
  - Limitation of re-arbitration and retransmission (1 to 7 or “unlimited” attempts).
  - Listen Only Mode.
  - Loop Back Mode (internal and external).
  - Transceiver Standby Mode.
- Extended status and error report:
  - Status report for frames, which are commanded to be transmitted.
  - Capturing of last occurred kind of error and of arbitration lost position.
  - Programmable Error Warning Limit.
- 32-bit AMBA APB Protocol Specification v2.0.
- Configurable interrupt sources.
- Single-port memory.
- Time-stamping:
  - CiA 603 time-stamping, external 16-bit counter.
- Compatible to AUTOSAR.
- Optimized for SAE J1939.

29.3 Functional Description

29.3.1 Block Diagram

Figure 29-1 shows the block diagram of the Controller Area Network (CAN).

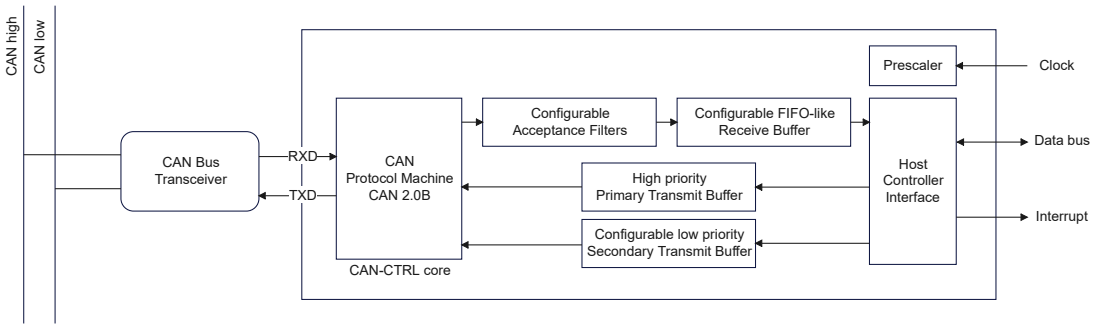


Figure 29-1 CAN Block Diagram

29.3.2 CAN Protocol

The CAN communication is organized in frames. Error frames and overload frames are handled fully automatically by the CAN2.0B module. Data frames are used to transmit data content from the host application.

There are several types of data frames. Classic CAN is able to transmit up to 8 bytes of data payload using one data frame. The figure below shows the frame format.

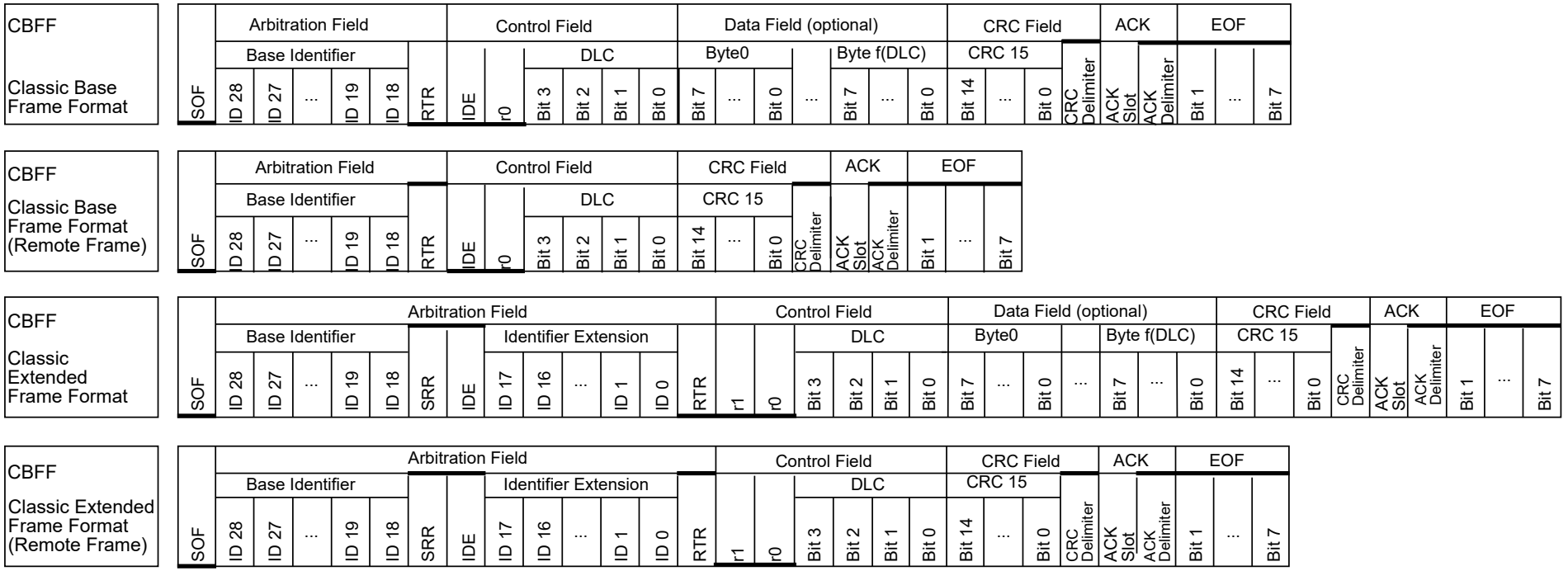


Table 29-1 CAN Bit Abbreviations

Abbreviation	Description	Comment
ID	Identifier	
RTR	Remote Transmission Request	Remote or data frame

Abbreviation	Description	Comment
IDE	Identifier Extension	Standard or extended frame
DLC	Data Length Code	Number of payload bytes
r1,r0,res	Reserved bits	

All CAN nodes have equal bus access; there is no super-node.

Data addressing is achieved through frame identifiers. In a CAN network, only one node can transmit frames with a specific identifier. All nodes receive all frames, and the host controller of a node must determine if a received frame is of interest. To lighten the load on a host controller, a CAN node can utilize acceptance filters. These filters compare received frame identifiers to user-selectable bit patterns. Only frames that pass an acceptance filter are stored in the receive buffer and signaled to the host controller.

The identifiers of CAN frames are used for bus arbitration. A CAN protocol machine stops the transmission of a frame with a low-priority identifier when a frame with a higher priority identifier is transmitted by another CAN node simultaneously. The CAN protocol machine automatically attempts to retransmit the stopped frame at the next possible transmit position.

29.3.3 Frame Buffers

Figure 29-2 shows the CAN2.0B module frame buffer block diagram.

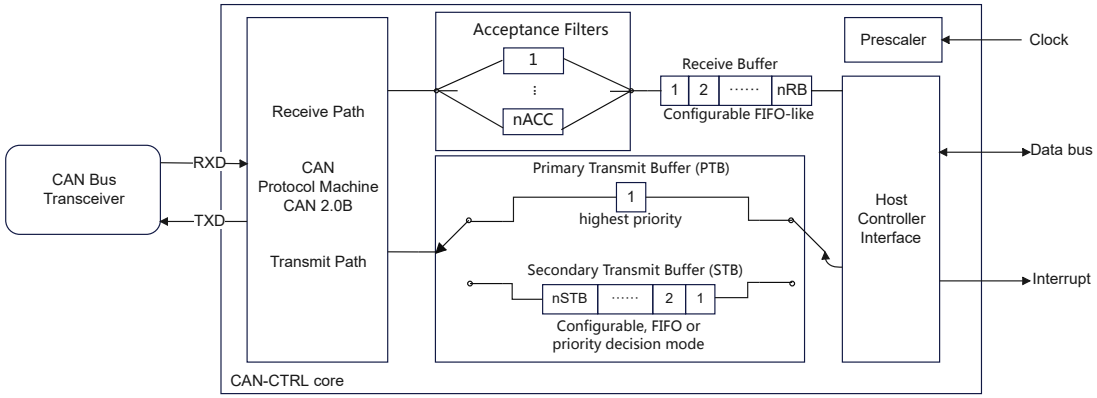


Figure 29-2 Frame Buffer Concept

29.3.3.1 Receive Buffer (RB)

To reduce the processing load on the host controller for received frames, acceptance filters are used. The CAN controller checks many bits of the frame header during acceptance filtering. If the received frame matches the filter criteria of one of the acceptance filters, then it will be stored in the Receive Buffer (RB), which has FIFO-like behavior.

Depending on the number of available RB slots, the host controller does not need to read received frames immediately. The CAN controller can generate interrupts upon every received frame when the RB is full or filled to a user-selectable “almost full” limit. Because of the FIFO-like behavior, the host controller always reads the oldest frame from the RB.

The host controller is not required to immediately read received frames, depending on the number of available RB slots. The CAN controller can generate interrupts when the RB is almost full or completely full, allowing space for frame reception. The host controller reads the oldest frame from the RB due to its behavior resembling a FIFO.

29.3.3.2 Transmit Buffer (TB)

For frame transmission purposes, two Transmit Buffers (TB) are offered. The Primary TB (PTB) has a higher priority, but is able to buffer only one frame. The Secondary TB (STB) has a lower priority. It can act in FIFO or in priority mode. The PTB has always higher priority than the STB and this is fully independent from the CAN bus arbitration. Bus arbitration is a priority decision based on the frame identifiers.

The STB can be commanded to transmit one or all stored frames. In FIFO mode the oldest frame inside this buffer is transmitted first. In priority mode the frame with the highest priority inside this buffer (based on the frame identifier) is transmitted first.

A PTB transmission stops and delays an STB transmission. The STB transmission is automatically restarted after the PTB frame has been successfully transmitted. A transmission starts at the next transmit position that is possible by the CAN protocol (after the next interframe space). Because of this, an STB transmission that has won the arbitration and is currently active, will be completed before a PTB transmission is started.

Interrupting STB transmissions using a PTB transmission may happen in the following cases:

- The STB is commanded to output all stored frames and the host controller decides to command a PTB transmission before all STB transmissions are completed.

- The STB is commanded to output a single frame and the host controller decides to command a PTB transmission before the STB transmission is started or before re-transmission or re-arbitration takes place.

If the host controller waits until each commanded transmission is completed, then it can easily decide which buffer shall transmit the next frame. As a drawback a frame with a low-priority identifier may block more important frames (refer to Aborting Transmission). Then the host could abort the frame. Another option would be to use automatic priority reordering (refer to TB Application Example in Priority Mode).

29.3.3.3 TB Application Example in FIFO Mode

Figure 29-3 shows the application example block diagram.

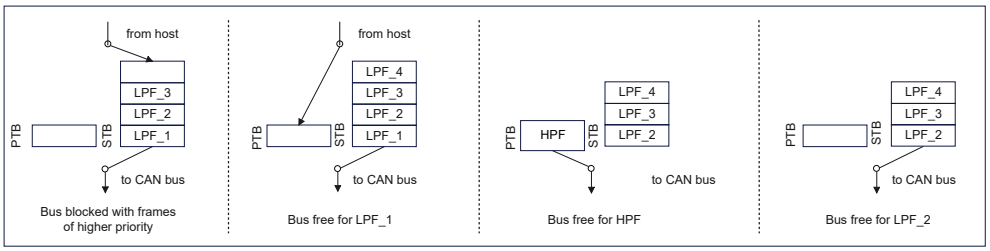


Figure 29-3 Transmit Buffer Application Example (TSALL = 1)

The given application example is explained step by step in Figure 29-3. In this figure, LPF_x is the abbreviation of “low priority frame” and HPF means “high priority frame”. As can be seen, LPF_1 is blocked by other higher priority frames from other CAN nodes in the first part of the figure. The host controller puts a fourth frame into the STB (LPF_4) and then decides to output the HPF. Meanwhile, the CAN bus was free to transmit LPF_1 which is followed by HPF, LPF_2 and so on.

The priority decision between the PTB and STB is done by the CAN protocol machine, but the CAN protocol itself uses another independent priority decision mechanism which is called bus arbitration. For bus arbitration the frame identifier defines the priority level.

In the example given above, it would be possible to place a frame with a very low-priority identifier in the PTB while higher priority frames remain in the STB. For the decision between PTB and STB always the PTB wins regardless of the frame identifier. It is the task of the host application to place frames with meaningful identifier priorities in the appropriate buffers.

29.3.3.4 TB Application Example in Priority Mode

In priority mode for the STB CAN controller automatically changes the order of the frames inside the STB. As a result, the frame with the highest priority is transmitted first. The priority decision is the same as for bus arbitration and therefore a frame identifier with a lower value has a higher priority. Regardless of that the PTB has always the highest priority.

Reordering the frames inside the STB is done automatically as soon as new frame is prepared for transmission and written into the STB. As a result, the frame with the highest priority is selected for transmission. For the host controller it is like fire-and-forget: it just needs to write a new frame into the STB while priority mode is active.

In general, a frame with a higher priority will never interrupt an active transmission. To give an example in Figure 29-3 in the leftmost part the frame LPF_1 is currently under transmission while LPF_4 is written to the STB. Even if LPF_4 has higher priority than LPF_1, still LPF_1 will be transmitted until it loses arbitration on the bus. Then after the node has lost arbitration, LPF_4 will be selected for transmission because of its priority.

Priority reordering is automatically done in background but takes some time. During heavy load when a lot of new frames with different priorities are written to the STB and the node loses arbitration or

there are errors on the bus, it may be that not the frame with the highest priority is selected. This may happen if this frame has been written too shortly before the deadline for the decision which frame will be next. But this is a very rare condition and in general the host application should not care about this.

If two frames with the same priority are written to the STB, then the oldest frame will be transmitted first regardless of any reordering with other lower- or higher-prioritized frames.

29.3.3.5 Aborting Transmission

If the situation arises, where a frame in a transmit buffer cannot be sent due to its low priority, this would block the buffer for a long time. To avoid this, the host controller can withdraw the transmission request and abort the frame. Aborting is possible for PTB and STB as well as for single or all frame transmissions.

29.3.3.6 A Full STB

After writing a frame to the STB, TSNEXT = 1 marks a buffer slot filled and jumps to the next free frame slot. TSNEXT is automatically reset to 0 by CAN-CTRL after this operation.

If the last frame slot has been filled and therefore all frame slots are occupied then TSNEXT stays set until a new frame slot becomes free. While TSNEXT = 1, then writing to TBUF is blocked by CAN-CTRL.

29.3.3.7 Error Handling

On one hand, CAN-CTRL does automatic error handling which means that in most cases the host controller does not need to care about errors. This includes automatic frame retransmission and automatic deletion of received frames with errors. On the other hand, if required by the host, CAN-CTRL may optionally give detailed information about errors and signal every error to the host by interrupt. This enables to run an application like a CAN bus monitor at the host.

Every CAN node has 3 states of error handling:

1. Error Active: The node automatically transmits active error frames upon detection of an error.
2. Error Passive: The node transmits a passive error frame upon detection of an error. This means that it does not transmit a dominant value to the bus, but expects an error frame transmitted by other CAN nodes.

   Error-passive nodes are allowed to transmit frames, but if they have been the transmitter of the previous frame they are required to suspend a further frame transmission for 8 bits following the intermission.
3. Bus Off: After too many errors a node goes “bus off” where it stops touching the bus.

29.3.3.8 Bus Off State

The BUSOFF bit in the CAN_CMD register is used to signal the bus off state. If a CAN node's transmit error counter exceeds 255, it will automatically enter the bus off state and will not participate in further communications until it returns to the error-active state. Enabling BUSOFF to 1 also sets the EIF to interrupt if EIE is enabled. To return to an error-active state, a CAN node must be reset by a power-on reset or receive 128 sequences of 11 recessive bits (recovery sequences).

NOTE: The bus may have a dominant state between each recovery sequence. Additionally, the minimum time between two valid frames is sufficient to be recognized as a recovery sequence.

29.3.3.9 Extended Status and Error Report

During CAN bus communication errors may occur. The following features support detection and analysis of them. This can be used for extended bus monitoring.

29.3.3.9.1 Programmable Error Warning Limit

Errors during reception / transmission are counted by RECNT and TECNT. A programmable error warning limit EWL, located in register CAN_ERRSTATS, can be used by the host controller for flexible reaction on those events. The limit values can be chosen in steps of 8 errors from 8 to 128:

Limit of count of errors = (EWL + 1) * 8

The interrupt EIF will be set if enabled by EIE under the following conditions:

- The border of the error warning limit has been crossed in either direction by RECNT or TECNT
- The BUSOFF bit has been changed in either direction.

29.3.3.9.2 Arbitration Lost Capture (ALC)

The loss of arbitration is not an error and therefore not relevant for TECNT / RECNT but also unwanted. CAN-CTRL is able to detect the exact bit position in the Arbitration Field where the arbitration has been lost. This event can be signaled by the ALIF interrupt if it is enabled by ALIE = 1. The value of ALC stays unchanged until another event of arbitration lost occurs.

29.3.3.9.3 Kind of Error (KOER)

CAN-CTRL recognizes errors on the CAN bus and stores the last error event in the KOER bits. A CAN bus error can be signaled by the BEIF interrupt if it is enabled with BEIE = 1. Every new error event overwrites the previous stored value of KOER. Therefore the host controller has to react quickly on the error event.

29.3.3.9.4 Reception of All Data Frames (RBALL)

If RBALL = 1, then all received data frames are stored, even those with errors. This also holds for loopback mode. Also, acceptance filtering is disabled. Only data frames are stored in RBUF. Error frames or overload frames are not stored.

If CiA 603 time-stamping is enabled (TSEN = 1) and the timestamp is configured for the EOF (TSPOS = 1), then in the case of an error, the time-stamp is acquired at the start of the error frame.

Most of the possible errors can only occur if the node is the transmitter of the frame. In this case, a frame would only be stored in RBUF if a loopback mode is activated. Depending on the type of error, parts of the frame stored in an RBUF slot may be valid, while other parts are unknown. Table 29-2 lists the possibilities

Table 29-2 RBALL and KOER

KOER	Condition	Description
NO ERROR	All	Successful reception.
BIT ERROR	Receiver	Only occurs in the ACK slot. All stored data are valid.
	Transmitter	The payload is always invalid. The header bits, including the ID, may be valid. In the arbitration phase, the detection of a wrong bit is part of the arbitration and therefore not a BIT ERROR. But if a stuff bit in the arbitration phase is detected wrong by the transmitter of the frame, then this is a BIT ERROR, and in this case, the header bits are invalid. Therefore, the header bits cannot be trusted, but if the header matches to an expected header of a frame, it can be used for further investigation.

KOER	Condition	Description
FORM ERROR	All	Only FORM ERRORs in a data frame are covered. This includes errors in the CRC delimiter, the ACK delimiter or the EOF bits. All stored data are valid.
STUFF ERROR	Receiver	The position of a STUFF ERROR is unknown. All stored data are invalid.
	Transmitter	Only occurs during arbitration. All stored data are invalid.
ACK ERROR	Receiver	Only occurs if the node is in LOM. All stored data are valid.
	Transmitter	Only occurs in loop-back mode without self-ACK. All stored data are valid.
CRC ERROR	Receiver	All stored data are invalid.
PCRC_ERROR	Receiver	All stored data are invalid.

NOTE: Although Table 29-2 indicates that the data may be valid under certain conditions, caution is necessary. The validity of the data is not guaranteed, and it depends on the specific error causing the issue. For instance, if the node has an incorrect bit timing configuration, there will be no proper synchronization and thus no valid data. Therefore, the statement that some or all of the data is valid merely suggests the potential for validity.

29.3.3.9.5 Transmit Status (TSTAT_1 and TSTAT_2)

There are different status and interrupt bits that can be used to monitor the status of frame transmissions, such as TPIF, ALIF, AIF, BEIF, and others. Furthermore, the transmit status registers TSTAT_1 and TSTAT_2 also provide comprehensive information.

TSTAT_1 contains data regarding the currently transmitted frame, making its content highly volatile. When a frame transmission is successfully finished or halted due to reaching the re-arbitration limit or retransmission limit (specified in CAN_CLKCTRL register), this information is saved in TSTAT_2. In simpler terms, TSTAT_2 is updated when a frame reaches its final state.

A transmission status provides the following details:

- A handle to identify the frame
- The transmission status of the frame identified by the handle
- The corresponding timestamp of the frame identified by the handle

The TTS register provides a timestamp that is updated only when the TTSEN bit in the LLC frame is set to 1. The TTS register is only relevant to the TSTAT_2 register. TSTAT bits within TSTAT_1 and TSTAT_2 encode frame status as specified inTable 29-3. The HANDLE identifier, which is chosen by the host application and included in the LLC frame, is used to identify a frame. It is recommended that a software counter value be used as the HANDLE, which is incremented with each new frame to be transmitted. This counter may roll over.

Table 29-3 TSTAT Status Encoding

Value	Label	Description
000	IDLE	No transmission in progress

Value	Label	Description
001	ONGOING	Transmission active without any issues
010	LOST_ARBITRATION	Arbitration lost. Re-arbitration may take place with respect to REALIM.
011	TRANSMITTED	Transmission successfully completed
100	ABORTED	Transmission aborted (TPA, TSA)
101	DISTURBED	Transmission error. Retransmission may take place with respect to RETLIM.
110	REJECTED	Misconfiguration of the frame format in the LLC frame.
111	-	Reserved

Several events may cause an update of TSTAT_1 or TSTAT_2, or both. Table 29-4 provides a comprehensive overview of these events.

Table 29-4 TSTAT Status Events

Event	TSTAT_1	TSTAT_2
Power-On reset or RESET	IDLE	IDLE
Start of a transmission (data fetch from TBUF by CAN protocol machine)	ONGOING	-unchanged-
Successful completion of a transmission	IDLE	TRANSMITTED
Transmission aborted (TPA or TSA)	IDLE	ABORTED
Arbitration lost, re-arbitration limit REALIM not reached	LOST_ARBITRATION	-unchanged-
Arbitration lost, re-arbitration limit REALIM reached	IDLE	LOST_ARBITRATION
Error, retransmission limit RETLIM not reached	DISTURBED	-unchanged-
Error, retransmission limit RETLIM reached	IDLE	DISTURBED
Bus-off recovery	ONGOING	-unchanged-
Misconfiguration in the LLC frame detected at start of transmission	IDLE	REJECTED

All events that trigger an update of TSTAT_1 or TSTAT_2, except for a transmission abort, will occur sequentially. As a result, the host application can monitor these events. However, a transmission abort can be commanded at any time. Additionally, there are complex scenarios of aborts that cannot be properly signaled by TSTAT_1 and TSTAT_2:

- If the STB is filled with more than one frame, a TSALL transmission is chosen, and TSA is commanded, then several frames will be aborted. TSTAT_2 can only reflect the status of one frame.
- In the case where both PTB and STB are being used, if the PTB transmission is currently active while the STB transmission is aborted, the successful transmission of the frame from PTB and the

abort of the frame(s) from STB may occur almost simultaneously. This would require changing the content of TSTAT_2 twice in a row, which the host application cannot track.

- If the STB is used, a TSALL transmission is chosen, and TSA is set short after one frame has been successfully completed and before the next one is started. This would require changing the content of TSTAT_2 twice in a row, and the host application cannot keep track of this.

Consequently, the status ABORTED will only be set for TSTAT_2 if all of the following conditions are true:

- TPA is set while a PTB transmission is active, or TSA is set while an STB transmission is active.
- The transmission currently being active has the status ONGOING, LOST_ARBITRATION or DISTURBED signaled by TSTAT_1.

29.3.3.9.6 Frame Rejection

If a frame is commanded to be transmitted and the frame format of this frame is misconfigured, then the frame will be rejected and not be transmitted.

Frame rejection will result in setting TPIF or TSIF (depending on if the PTB or STB was used and if the appropriate interrupt enable bit was set) but will be signaled using the transmit status REJECTED as defined in Transmit Status (TSTAT_1 and TSTAT_2) .

The reason for a frame format misconfiguration is a serious error in an upper application layer and shall not happen during normal operation. It is expected to happen only during application development because of mistakes.

29.3.3.10 Extended Features

29.3.3.10.1 Retransmission and Re-arbitration Limitation

Sometimes, it may not be desirable to have automatic retransmission or re-arbitration, as repeated attempts could cause significant delays and increase the bus load. RETLIM and REALIM can both be used to limit these issues independently. Retransmission occurs after an error, while re-arbitration occurs when arbitration is lost. Retransmission may be halted if the bus-off state is reached.

The retransmission and re-arbitration counters will be incremented when the corresponding events occur. After a successful transmission, both counters will be reset to 0. It is irrelevant whether the PTB or the STB was chosen as the source of the transmission. Only the events of error, arbitration loss, or successful transmission affect the two counters. The following example explains this behavior in a step-by-step manner:

1. If the STB transmission fails due to an error, the retransmission counter is incremented.
2. If the STB transmission loses arbitration, the re-arbitration counter is incremented.
3. If the PTB transmission is successful, both counters are reset (the frame in the STB is still not transmitted).

In the event of a successful transmission, there is no distinction from normal transmission. However, if the transmission is unsuccessful, the following actions will occur:

- If enabled, TPIF will be set and the corresponding transmit buffer slot will be cleared.
- If there is an error, KOER and the error counters will be updated. If BEIE is enabled, BEIF will be set, and the other error interrupt flags will respond accordingly.
- If there is a lost arbitration, ALIF will be set if ALIE is enabled.

Therefore, if the limitations on retransmission or re-arbitration are in place, TPIF alone cannot indicate whether the frame has been successfully transmitted. Therefore, if a feedback regarding the success of a transmission is required, retransmission or re-arbitration limitations should only be used

in conjunction with BEIF and ALIF. Additionally, using TSTAT_1 and TSTAT_2 can help keep track of the events.

If retransmission or re-arbitration is limited and TSALL = 1 is set, and there is more than one frame in the STB, the attempts will be counted for each frame. If any of the frames are not successfully transmitted before reaching one of the limits, CAN-CTRL moves on to the next frame and eventually stops if the STB is empty. In this scenario, it is quite complex to determine what has occurred, and only the error counter TECNT indicates what has happened. Evaluating this can be challenging because the host cannot detect which of the two frames was the successful one if one of them had errors.

It is recommended to use retransmission or re-arbitration limitation only if successful transmissions are handled at a higher application layer, such as by using acknowledge messages replied by the receiver of a message. Limiting retransmission and re-arbitration allows for operation in time slots, ensuring that message delays are kept to a minimum.

29.3.3.10.2 Listen Only Mode (LOM)

LOM provides the ability of monitoring the CAN bus without any influence to the bus. LOM by CAN-CTRL is compatible to the bus monitoring feature defined in ISO 11898-1:2015.

Another application is the automatic bit rate detection where the host controller tries different timing settings until no errors occur.

Errors will be monitored (KOER, BEIF) in LOM.

In LOM CAN-CTRL is not able to write dominant bits onto the bus (no active error flags or overload flags, no acknowledge). This is done using the following rules:

- In LOM the protocol machine acts as if in error passive mode where only recessive error flags are generated. Only the protocol machine acts as if in error passive mode. All other components including the status registers are not touched.
- In LOM the protocol machine does not generate a dominant ACK.
- The error counters are not modified in any direction.

Important facts regarding ACKs for LOM:

- If a frame is transmitted by a node then an ACK visible at the bus will be only generated if at least one additional node is attached to the bus and permitted to generate the ACK. Then there will be no error and all nodes (also those in LOM) will receive the frame. Otherwise an ACK error invalidates the frame.
- If there is an active or passive error flag after an ACK error, then the node in LOM is able to detect this as ACK error.

Activation of LOM cannot be done while a transmission is active. If LOM is enabled then no transmission can be started.

The loop back mode (external) LBME plays an important role for the behavior of LOM. If LBME is disabled, then LOM acts as described above and the node cannot write any dominant bit to the bus. But if LBME is activated, then the node is allowed to transmit a frame. LBME allows only the transmission of a frame including the optional self-ACK (SACK), but the node will not respond with an ACK to frames from other nodes and will not generate error or overload frames. In Summary the combination of LOM and LBME is “a silent receiver, which is able to transmit if necessary”.

29.3.3.10.3 Restricted Operation (ROP)

A CAN node which is in restricted operation mode shall not transmit data frames unless it is a potential time master. Then such a node shall be able to transmit time reference messages to start up the network.

CAN-CTRL does not provide any protection to block transmissions. The host application is fully responsible follow these transmission rules.

A CAN node which is in restricted operation mode is able to receive frames and will respond with ACK to valid frames. In case of an error or overload condition the node with ROP = 1 will treat this as protocol exception and will enter bus integration. The error counters are not modified in any direction if ROP = 1.

Activation of ROP cannot be done while a transmission is active. If ROP is enabled then no transmission can be started.

29.3.3.10.4 Bus Connection Verification

To check if a node is connected to the bus, follow these steps:

1. Limit retransmission to one single shot (RETLIM = 0).
2. Transmit a frame.

   If the node is connected to the bus, its TX bits will be visible on its RX input.
3. If there are other nodes connected to the CAN bus, a successful transmission (including an acknowledge from the other nodes) is expected, and no error will be indicated.
4. If the node is the only one connected to the CAN bus (with a proper connection between the bus, the transceiver, and CAN-CTRL), the first regular error situation occurs in the ACK slot because of no acknowledgement from other nodes. In this case, a BEIF error interrupt will be generated if enabled and KOER="100" (ACK error).
5. If the connection to the transceiver or the bus is broken, the BEIF error interrupt will be set immediately after the SOF bit, and KOER="001" (BIT error).

29.3.3.10.5 Loopback Modes

CAN-CTRL supports two loopback modes, both of which allow for receiving the transmitted frame, which can be useful for self-tests:

- **Loopback mode, internal (LBMI)**

  In LBMI, CAN-CTRL is disconnected from the CAN bus and the <txd> output is set to recessive. The output data stream is internally fed back to the input. In LBMI, the node generates a self-ACK to avoid an ACK error.
- **Loopback mode, external (LBME)**

  In LBME, CAN-CTRL stays connected to the transceiver and a transmitted frame will be visible on the bus. With the help of the transceiver, CAN-CTRL receives its own frame. In LBME, the node does not generate a self-ACK when SACK = 0, but generates a self-ACK when SACK = 1. Therefore, in LBME with SACK = 0, there are two possible results upon a frame transmission:

  1. Another node also receives the frame and generates an ACK, leading to successful transmission and reception.
  2. If no other node is connected to the bus, an ACK error occurs. To prevent retransmissions and avoid incrementing the error counters, it is recommended to set RETLIM = 0 when unsure about the presence of other connected nodes.

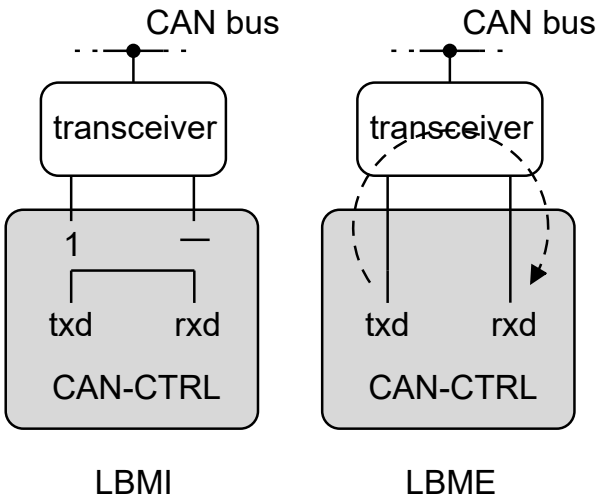


Figure 29-4 LBMI and LBME

In loopback mode, CAN-CTRL receives its own frame, stores it in the RBUF and sets the appropriate receive and transmit interrupt flags if enabled. Own received frames are flagged with LBF = 1.

LBMI can be useful for chip-internal tests and software tests, while LBME can test the transceiver and its connections. LBMI and LBME should not be changed together with TPE, TSONE or TSALL.

If the node is connected to a CAN bus, switching back from LBMI to normal operation must not be done by simply setting LBMI to 0. This is because it may occur at the same time as another CAN node is transmitting. In this case, switching back to normal operation should be done by setting the RESET bit to 1. This automatically clears LBMI to 0. Finally, RESET can be disabled, and CANCTRL returns to normal operation. On the other hand, LBME can be disabled at any time. LBME can be used in combination with LOM.

Receiving the own transmitted frame triggers the appropriate transmit and receive interrupts. The receive interrupt will be set earlier because a frame becomes valid for a receiver at the last but one bit of EOF, but for a transmitter at the last bit of EOF.

29.3.3.10.6 Transceiver Standby Mode

By utilizing the STBY bit, the output signal <stby> is activated, enabling the activation of a standby mode for a transceiver. This functionality is in line with the NXP TJA1049 transceiver and other transceivers that offer a comparable feature.

Once standby is activated, no transmission is possible. Therefore, TPE, TSONE, and TSALL cannot be set. On the other hand, CAN-CTRL does not allow STBY to be set if a transmission is already active (TPE, TSONE, or TSALL is set).

When STBY is activated, the transceiver goes into a low-power state where it can't receive a frame at full speed. However, it still monitors the CAN bus for a dominant state. If this state is active for a duration specified in the transceiver's data sheet, the transceiver will pull the <rxd> signal low. Once <rxd> is low, CANCTRL will automatically reset STBY to 0, disabling standby mode for the transceiver. This process happens quietly without any interruption to the host controller. (It's assumed that when leaving standby mode, a frame from another node will be received.)

Switching from standby mode to active mode takes some time for the transceiver and therefore the initial wakeup frame cannot be received successfully. Therefore, the node recently being in standby will not respond with an ACK. If no CAN node at the bus responds to the wakeup frame with an ACK, then this results in an ACK error for the transmitter of the wakeup frame. Then, the transmitter will

automatically retransmit the frame. At the time of the retransmission, the transceiver will be back in active mode, and CAN-CTRL will receive the frame and will respond with an ACK.

In summary, one node transmits a wakeup frame. If all other nodes are in standby mode, the transmitter receives an ACK error and will automatically resend the frame. During the resend, the nodes switch back to active mode and respond with an ACK.

29.3.3.10.7 Error Counter Reset

According to the CAN standard, RECNT counts receive errors, while TECNT counts transmit errors. If a CAN node encounters excessive transmit errors, it will enter the bus off state (refer to Error Handling). The bit RESET does not alter the error counters or the bus off state. The CAN specification provides guidelines for exiting the bus off state and decreasing the error counters. A properly functioning node will automatically recover from these problems if they are due to temporary errors. The classic CAN 2.0B specification requires this automatic behavior, eliminating the need for host controller interaction to prevent communication disruption.

Resetting the error counters is achieved by writing a 1 to the bit BUSOFF, which subsequently causes the node to exit the bus off state. This operation is performed without setting EIF.

29.3.3.10.8 Software Reset

If the bit RESET in CFG_STAT is set to 1, then the software reset is activated. While RESET = 1, several components are forced into a reset state, but not all components are affected by RESET. Some components are only responsive to a hardware reset. The reset values of all bits are always the same for both software and hardware resets.

Table 29-5 Software Reset

Register	RESET	Comment
ACFADR	No	-
ACFC	No	The register is writable when RESET = 1 and write-locked when 0.
ACFM	No	The register is writeable when RESET = 1 and write-locked when 0.
AC_SEG_1	No	The register is writeable when RESET = 1 and write-locked when 0.
AC_SEG_2	No	The register is writeable when RESET = 1 and write-locked when 0.
AC_SJW	No	The register is writeable when RESET = 1 and write-locked when 0.
AE_x	No	-
AFWL	No	-
AIF	Yes	-
ALC	Yes	-
ALIE	No	-
ALIF	Yes	-
BEIE	No	-

Register	RESET	Comment
BEIF	Yes	-
BUSOFF	(No)	An error counter reset by setting BUSOFF = 1 also resets BUSOFF.
EIE	No	-
EIF	No	-
EPASS	No	-
EPIE	No	-
EPIF	Yes	-
EWARN	No	-
EWL	Yes	-
KOER	Yes	-
LBME	Yes	-
LBMI	Yes	-
LOM	No	-
PRESC	No	The register is writeable when RESET = 1 and write-locked when 0.
RAFIE	No	-
RAFIF	Yes	-
RBALL	Yes	-
RBUF	(Yes)	All RB slots are marked as empty. RBUF contains unknown data.
REALIM	Yes	-
RECNT	No	An error counter reset is possible by setting BUSOFF = 1.
RETLIM	Yes	-
RFIE	No	-
RFIF	Yes	-
RIE	No	-
RIF	Yes	-
ROIE	No	-
ROIF	Yes	-
ROM	No	-
ROP	No	-
ROV	Yes	All RB slots are marked as empty.
RREL	Yes	-

Register	RESET	Comment
RSTAT	Yes	All RB slots are marked as empty.
SACK	Yes	-
STBY	No	-
TBSEL	Yes	TBUF fixed to point to PTB.
TBUF	(Yes)	All STB slots are marked as empty. Because of TBSEL TBUF points to the PTB.
TECNT	No	An error counter reset is possible by setting BUSOFF = 1.
TPA	Yes	-
TPE	Yes	-
TSA	Yes	-
TSALL	Yes	-
TSMODE	No	-
TSEN	No	-
TSPOS	No	-
TSNEXT	Yes	-
TSONE	Yes	-
TPIE	No	-
TPIF	Yes	-
TSFF	Yes	All STB slots are marked as empty.
TSIE	No	-
TSIF	Yes	-
TSSTAT	Yes	All STB slots are marked as empty.

29.3.4 Time-Stamping

The CAN in Automation (CiA) defines in specification CiA 603 a method for time-stamping with at least 16 bits, which CAN-CTRL optionally supports. The basic concept of CiA 603 is to have a free-running timer which counts clock cycles and not CAN bit times. The precision shall be 10µs (16-bit). Time stamps can be acquired at the SOF or EOF of a CAN frame.

CiA 603 is supposed to support time-stamping and time-synchronization of AUTOSAR. For AUTOSAR, one node in the CAN network is the time master. A time master transmits a synchronization message (SYNC message). The time-stamp of the SYNC message is acquired by the time master and all time slaves. The difference in time between the event of commanding a SYNC message until the time when the SYNC message actually gets transmitted will be transmitted in a follow-up (FUP) message by the time master.

CiA 603 defines rules to read out the timer and modify the timer. CAN-CTRL does not include the timer, but uses an external timer. CAN-CTRL only includes the mechanism of time-stamping, the register to store one transmission time stamp (TTS) and the memory to store reception time stamps (RTS) for all received messages.

29.3.5 Acceptance Filters (ACF)

To reduce the load of received frames for the host controller, CAN controller uses acceptance filters. CAN controller checkes the frame header during acceptance filters.

If a frame passes one of the filters then it will be accepted. If accepted, the frame will be stored into the RB and finally RIF is set if RIE is enabled. If the frame is not accepted, RIF is not set and the RB FIFO pointer is not increased. Frames that are not accepted will be discarded and overwritten by the next frame. No stored valid and accepted frame will be overwritten by any not accepted frame.

Acceptance filtering is only done for valid frames. (In case of errors during transmission CAN-CTRL will do error handling according to the CAN specification.)

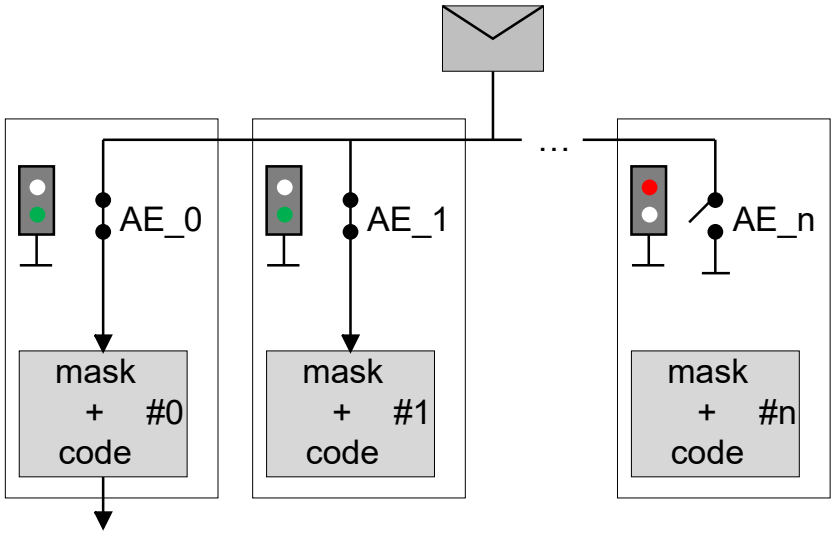


Figure 29-5 Example of Acceptance Filtering

29.4 Registers

29.4.1 Register Address Map

Offset	Register Name	Register Description
0x0000	CAN_CFG	Configuration register, including VER_0/1, CiA603 timestamp configuration
0x0004	CAN_BITTIME	Bit timing register
0x0010	CAN_CLKCTRL	Clock control register, including clock prescaler, re-arbitration and retransmission limit configuration
0x0014	CAN_INTF	Interrupt flag register
0x0018	CAN_INTE	Interrupt enable register
0x001c	CAN_TSTAT	Transmit status register
0x0020	CAN_TSCSR	Transmission time stamp register
0x0028	CAN_CMD	Command register, including configuration and status, transmission command, transmission control, receive control configuration
0x002c	CAN_ERRSTATS	Including warning limit, arbitration lost position, error and error counter
0x0044	CAN_ACF	Acceptance filter control and enable register
0x0048	CAN_ACFIDENTREG	Acceptance filter ID configuration
0x004c	CAN_ACFFORMAT	Acceptance filter frame format configuration
0x0058	CAN_ACMIDENTREG	Acceptance filter ID mask configuration
0x005c	CAN_ACMFORMAT	Acceptance filter frame format mask configuration
0x0070	CAN_RBID	Receive buffer identifier register
0x0074	CAN_RBFORMAT	Receive buffer frame format register
0x0080	CAN_RBTSREG	Receive buffer timestamp register

Offset	Register Name	Register Description
0x008c	CAN_RBDATAREG0	Receive buffer data0 register
0x0090	CAN_RBDATAREG1	Receive buffer data1 register
0x0890	CAN_TBID	Transmit buffer identifier register
0x0894	CAN_TBFORMAT	Transmit buffer frame format register
0x08ac	CAN_TBDATAREG0	Transmit buffer data0 register
0x08b0	CAN_TBDATAREG1	Transmit buffer data1 register

29.4.2 Register Field Details

29.4.2.1 CAN_CFG

0x0000			Configuration Register											CAN_CFG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved						TSPOS	TSEN	Reserved						ROP	CES
Type	RO						RW	RW	RO						RW	RW
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	VER1								VER0							
Type	RO								RO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-6 Configuration Register Description

Field	Name	Description
31:26	Reserved	Reserved
25	TSPOS	Time-stamping position 0: SOF 1: EOF TSPOS can only be changed if TSEN = 0, but it is possible to modify TSPOS with the same write access that sets TSEN = 1.
24	TSEN	Time-stamping enable 0: Disabled 1: Enabled
23:18	Reserved	Reserved
17	ROP	Restricted operation

Field	Name	Description
		0: Restricted operation disabled 1: Restricted operation enabled ROP cannot be changed while a transmission is active. No transmission can be started if ROP is enabled.
16	CES	CAN error signaling 0: Error signaling disabled 1: Error signaling enabled
15:8	VER1	Version of CAN-CTRL. VER_1 holds the major version and VER_0 the minor version
7:0	VER0	Version of CAN-CTRL. VER_1 holds the major version and VER_0 the minor version

29.4.2.2 CAN_BITTIME

0x0004			Bit timing register											CAN_BITTIME		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved	ACSJW							Reserved	ACSEG2						
Type	RO	RW							RO	RW						
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved							ACSEG1								
Type	RO							RW								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 29-7 Bit Timing Register Description

Field	Name	Description
31	Reserved	Reserved
30:24	ACSJW	Synchronization jump width
23:23	Reserved	Reserved
22:16	ACSEG2	Bit timing segment 2
15:9	Reserved	Reserved
8:0	ACSEG1	Bit timing segment 1

29.4.2.3 CAN_CLKCTRL

0x0010			Clock control register											CAN_CLKCTRL		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved	RETLIM			Reserved	REALIM			Reserved							
Type	RO	RW			RO	RW			RO							
Reset	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved											PRESC				
Type	RO											RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-8 Clock Control Register Description

Field	Name	Description
31	Reserved	Reserved
30:28	RETLIM	Retransmission limit 000: 1 attempt (no retransmission) ... 110: 7 attempts 111: Unlimited (only limited by the transmit error counter TECNT)
27	Reserved	Reserved
26:24	REALIM	Re-arbitration limit 000: 1 attempt (no re-arbitration) ... 110: 7 attempts 111: Unlimited

Field	Name	Description
23:5	Reserved	Reserved
4:0	PRESC	Prescaler

29.4.2.4 CAN_INTF

0x0014			Interrupt flag register											CAN_INTF		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EWARN	EPASS	Reserved													
Type	RO	RO	RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					EPIF	ALIF	BEIF	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF
Type	RO					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-9 Interrupt Flag Register Description

Field	Name	Description
31	EWARN	Error warning limit reached 0: The values in both counters are less than EWL. 1: One of the error counters, RECNT or TECNT, is equal to or bigger than EWL.
30	EPASS	Error passive status 0: Node is error active 1: Node is error passive
29:11	Reserved	Reserved
10	EPIF	Error passive interrupt flag EPIF will be activated if EPASS changes and if EPIE = 1. Write 1 to clear the flag. Writing a 0 has no effect.
9	ALIF	Arbitration lost interrupt flag

Field	Name	Description
		Write 1 to clear the flag; writing a 0 has no effect.
8	BEIF	Bus error interrupt flag Write 1 to clear the flag; writing a 0 has no effect.
7	RIF	Receive interrupt flag 0: No frame has been received. 1: Data or a remote frame has been received and is available in the receive buffer. Write 1 to clear the flag; writing a 0 has no effect.
6	ROIF	RB overflow interrupt flag 0: No RB overwritten. 1: At least one received frame has been overwritten in the RB. Write 1 to clear the flag; writing a 0 has no effect.
5	RFIF	RB full interrupt flag 0: The RB FIFO is not full. 1: All RBs are full. If no RB is released until the next valid frame is received, the oldest frame will be lost. Write 1 to clear the flag; writing a 0 has no effect.
4	RAFIF	RB Almost full interrupt flag 0: Number of filled RB slots is less than AFWL 1: Number of filled RB slots is greater than or equal to AFWL Write 1 to clear the flag; writing a 0 has no effect.
3	TPIF	Transmission primary interrupt flag 0: No transmission of the PTB has been completed. 1: The requested transmission of the PTB has been successfully completed. Write 1 to clear the flag; writing a 0 has no effect.
2	TSIF	Transmission secondary interrupt flag 0: No transmission of the STB has been completed successfully.

Field	Name	Description
		1: The requested transmission of the STB has been successfully completed. Write 1 to clear the flag; writing a 0 has no effect.
1	EIF	Error interrupt flag 0: There has been no change. 1: The border of the error warning limit has been crossed in either direction, or the BUSOFF bit has been changed in either direction. Write 1 to clear the flag; writing a 0 has no effect.
0	AIF	Abort interrupt flag 0: No abort has been executed. 1: After setting TPA or TSA, the appropriated frames have been aborted. It is recommended to not set both TPA and TSA simultaneously because of both source AIF. Write 1 to clear the flag; writing a 0 has no effect.

29.4.2.5 CAN_INTE

0x0018			Interrupt enable register											CAN_INTE		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					EPIE	ALIE	BEIE	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	Reserved
Type	RO					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0

Table 29-10 Interrupt Enable Register Description

Field	Name	Description
31:11	Reserved	Reserved
10	EPIE	Error passive interrupt enable
9	ALIE	Arbitration lost interrupt enable
8	BEIE	Bus error interrupt enable
7	RIE	Receive interrupt enable 0: Disabled 1: Enabled
6	ROIE	RB overflow interrupt enable 0: Disabled 1: Enabled

Field	Name	Description
5	RFIE	RB full interrupt enable 0: Disabled 1: Enabled
4	RAFIE	RB almost full interrupt enable 0: Disabled 1: Enabled
3	TPIE	Transmission primary interrupt enable 0: Disabled 1: Enabled
2	TSIE	Transmission secondary interrupt enable 0: Disabled 1: Enabled
1	EIE	Error interrupt enable 0: Disabled 1: Enabled
0	Reserved	Reserved

29.4.2.6 CAN_TSTAT

0x001c			Transmit status register											CAN_TSTAT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved					TSTAT2			HANDLE2							
Type	RO					RW			RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					TSTAT1			HANDLE1							
Type	RO					RW			RW							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-11 Transmit Status Register Description

Field	Name	Description
31:27	Reserved	Reserved
26:24	TSTAT2	Transmission Status Code: 000: IDEL, no transmission in progress. 001: ONGOING, transmission active without any issues. 010: LOST_ARBITRATION, arbitration lost, re-arbitration may take place with respect to REALIM. 011: TRANSMITTED, transmission successfully completed. 100: ABORTED, transmission aborted (TPA,TSA). 101: DISTURBED, transmission error, retransmission may take place with respect to RETLIM. 110: REJECTED, misconfiguration of the frame format. 111: Reserved.
23:16	HANDLE2	Handle for frame identification

Field	Name	Description
15:11	Reserved	Reserved
10:8	TSTAT1	Transmission Status Code: 000: IDEL, no transmission in progress. 001: ONGOING, transmission active without any issues. 010: LOST_ARBITRATION, arbitration lost, re-arbitration may take place with respect to REALIM. 011: TRANSMITTED, transmission successfully completed. 100: ABORTED, transmission aborted (TPA,TSA). 101: DISTURBED, transmission error, retransmission may take place with respect to RETLIM. 110: REJECTED, misconfiguration of the frame format. 111: Reserved.
7:0	HANDLE1	Handle for frame identification

29.4.2.7 CAN_TSCSR

0x0020			Transmission time stamp register											CAN_TSCSR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TTS															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-12 Transmission Time Stamp Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	TTS	Transmission Time Stamp (TTS) TTS holds the timestamp of the most recent transmitted frame for CiA 603 time stamping. When TTSEN = 1, each new frame overwrites the existing TTS. The purpose of TTS is to assist the time master in obtaining the timestamp of the SYNC message. If TSEN = 0, then TTS is set to 0.

29.4.2.8 CAN_CMD

0x0028			Command register											CAN_CMD		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	SACK	ROM	ROV	RREL	RBALL	Reserved	RSTAT		Reserved	TSNEXT	TSMODE	Reserved		TSFF	TSSTAT	
Type	RW	RW	RO	RW	RW	RO	RO		RO	RW	RW	RO		RW	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	RESET	LBME	LBMI	Reserved				BUSOFF
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RO				RW
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 29-13 Command Register Description

Field	Name	Description
31	SACK	Self-Acknowledge 0: No self-ACK 1: Self-ACK when LBME = 1
30	ROM	Receive buffer overflow mode In case of a full RBUF, when a new frame is received, ROM selects the following: 0: The oldest frame will be overwritten. 1: The new frame will not be stored.
29	ROV	Receive buffer overflow 0: No overflow 1: Overflow. At least one frame is lost.
28	RREL	Receive buffer release

Field	Name	Description
		0: No release 1: Release. The host has read the RB.
27	RBALL	Receive buffer stores all data frames 0: Normal operation 1: RB stores correct data frames as well as data frames with error
26	Reserved	Reserved
25:24	RSTAT	Receive buffer level status 00: Empty 01: Higher than empty and lower than almost full level (AFWL) 10: Higher than or equal to almost full level (programmable threshold by AFWL) but not full and not overflow 11: Full (stays set in case of overflow, for overflow signaling, see ROV)
23	Reserved	Reserved
22	TSNEXT	Transmit buffer secondary next 0: No action 1: STB slot filled, select next slot.
21	TSMODE	Transmit buffer secondary operation mode 0: FIFO mode 1: Priority decision mode
20:19	Reserved	Reserved
18	TSFF	Transmit secondary buffer full flag 0: The STB is not filled with the maximal number of frames. 1: The STB is filled with the maximal number of frames. If the STB is disabled using STB_ENABLE = 0, then TSFF = 0.
17:16	TSSTAT	Transmission secondary status bits

Field	Name	Description
		00: STB is empty 01: STB is less than or equal to half full 10: STB is more than half full 11: STB is full If the STB is disabled using STB_ENABLE = 0, then TSSTAT = 00.
15	TBSEL	Transmit buffer select 0: PTB (high-priority buffer) 1: STB
14	LOM	Listen only mode 0: Disabled 1: Enabled
13	STBY	Transceiver standby mode 0: Disabled 1: Enabled
12	TPE	Transmit primary enable 0: No transmission for the PTB 1: Transmission enable for the frame in the high-priority PTB
11	TPA	Transmit primary abort 0: No abort 1: Aborts a transmission from PTB, which has been requested by TPE = 1 but has not started yet. (The data bytes of the frame remain in the PTB.)
10	TSONE	Transmit secondary one frame 0: No transmission for the STB 1: Transmission enables one in the STB
9	TSALL	Transmit secondary all frames 0: No transmission for the STB

Field	Name	Description
		1: Transmission enables all frames in the STB
8	TSA	Transmit secondary abort 0: No abort 1: Aborts a transmission from STB, which has been requested but not started yet
7	RESET	Reset request bit 0: No local reset of CAN-CTRL 1: The host controller performs a local reset of CAN-CTRL
6	LBME	Loopback mode, external 0: Disabled 1: Enabled
5	LBMI	Loopback mode, internal 0: Disabled 1: Enabled
4:1	Reserved	Reserved
0	BUSOFF	Bus off (bus status bit) 0: The controller status is 'bus on'. 1: The controller status is 'bus off'.

29.4.2.9 CAN_ERRSTATS

0x002c			Error status register											CAN_ERRSTATS		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TECNT								RECNT							
Type	RO								RO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	KOER		ALC						AFWL				EWL			
Type	RO		RO						RW				RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1

Table 29-14 Error Status Register Description

Field	Name	Description
31:24	TECNT	Transmit error count (number of errors during transmission)
23:16	RECNT	Receive error count (number of errors during reception)
15:13	KOER	Kind of error (error code) 000: No error 001: Bit error 010: Form error 011: Stuff error 100: Acknowledgement error 101: CRC error 110: Other error (Dominant bits after own error flag, received active error flag too long, dominant bit during Passive-Error-Flag after ACK error)

Field	Name	Description
12:8	ALC	Arbitration lost capture (bit position in the frame where the arbitration has been lost)
7:4	AFWL	Receive buffer almost full warning limit
3:0	EWL	Programmable error warning limit

29.4.2.10 CAN_ACF

0x0044			Acceptance filter control and enable register											CAN_ACF		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved												AE3	AE2	AE1	AE0
Type	RO												RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved														ACFADR	
Type	RO														RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-15 Acceptance Filter Control and Enable Register Description

Field	Name	Description
31:20	Reserved	Reserved
19	AE3	Acceptance filter3 enable 0: Acceptance filter disabled 1: Acceptance filter enabled
18	AE2	Acceptance filter2 enable 0: Acceptance filter disabled 1: Acceptance filter enabled
17	AE1	Acceptance filter1 enable 0: Acceptance filter disabled 1: Acceptance filter enabled
16	AE0	Acceptance filter0 enable 0: Acceptance filter disabled

Field	Name	Description
		1: Acceptance filter enabled
15:2	Reserved	Reserved
1:0	ACFADR	Acceptance filter address

29.4.2.11 CAN_ACFIDENTREG

0x0048			Acceptance filter ID configuration register											CAN_ACFIDENTREG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		ACFID													
Type	RO		RW													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ACFID															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-16 Acceptance Filter ID Configuration Register Description

Field	Name	Description
31:29	Reserved	Reserved
28:0	ACFID	Frame identifier

29.4.2.12 CAN_ACFFORMAT

0x004c			Acceptance filter frame format configuration Register											CAN_ACFFORMAT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved			ACFLBF	ACFESI	Reserved					ACFSEC	ACFRMF	Reserved			ACFIDE
Type	RO			RW	RW	RO					RW	RW	RO			RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												ACFDLC			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-17 Acceptance Filter Frame Format Configuration Register Description

Field	Name	Description
31:29	Reserved	Reserved
28	ACFLBF	Loop-back frame
27	ACFESI	Error state indicator
26:22	Reserved	Reserved
21	ACFSEC	Simple or extended content 0: Simple content 1: Extended content
20	ACFRMF	Remote Frame 0: Data frame 1: Remote frame
19:17	Reserved	Reserved

Field	Name	Description
16	ACFIDE	Identifier extension 0: Standard format: ID(28:18) 1: Extended format: ID(28:0)
15:4	Reserved	Reserved
3:0	ACFDLC	Data length code

29.4.2.13 CAN_ACMIDENTREG

0x0058			Acceptance filter ID mask configuration register											CAN_ACMIDENTREG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		ACMID													
Type	RO		RW													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ACMID															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-18 Acceptance Filter ID Mask Configuration Register Description

Field	Name	Description
31:29	Reserved	Reserved
28:0	ACMID	Frame identifier

29.4.2.14 CAN_ACMFORMAT

0x005c			Acceptance filter frame format configuration register											CAN_ACMFORMAT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved			ACMLBF	ACMESI	Reserved					ACMSEC	ACMRMF	Reserved			ACMIDE
Type	RO			RW	RW	RO					RW	RW	RO			RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												ACMDLC			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-19 Acceptance Filter Frame Format Configuration Register Description

Field	Name	Description
31:29	Reserved	Reserved
28	ACMLBF	Loop-back frame
27	ACMESI	Error state indicator
26:22	Reserved	Reserved
21	ACMSEC	Simple or extended content 0: Simple content 1: Extended content
20	ACMRMF	Remote frame 0: Data frame 1: Remote frame

Field	Name	Description
19:17	Reserved	Reserved
16	ACMIDE	Identifier extension 0: Standard format: ID(28:18) 1: Extended format: ID(28:0)
15:4	Reserved	Reserved
3:0	ACMDLC	Data length code

29.4.2.15 CAN_RBID

0x0070			Receive buffer identifier register											CAN_RBID		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved		RBID													
Type	RO		RW													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RBID															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-20 Receive Buffer Identifier Register Description

Field	Name	Description
31:29	Reserved	Reserved
28:0	RBID	Frame identifier

29.4.2.16 CAN_RBFORMAT

0x0074			Receive buffer frame format register											CAN_RBFORMAT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved			RBLBF	RBESI	RKOER			Reserved		RBSEC	RMF	Reserved			RBIDE
Type	RO			RW	RW	RO			RO		RW	RW	RO			RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												DLC			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-21 Receive Buffer Frame Format Register Description

Field	Name	Description
31:29	Reserved	Reserved
28	RBLBF	Loop-back frame
27	RBESI	Error state indicator
26:24	RKOER	Kind of error
23:22	Reserved	Reserved
21	RBSEC	Simple or extended content 0: Simple content 1: Extended content
20	RMF	Remote frame 0: Data frame 1: Remote frame

Field	Name	Description
19:17	Reserved	Reserved
16	RBIDE	Identifier extension 0: Standard format: ID(28:18) 1: Extended format: ID(28:0)
15:4	Reserved	Reserved
3:0	DLC	Data length code

29.4.2.17 CAN_RBTSREG

0x0080			Receive buffer timestamp register											CAN_RBTSREG		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RBRTS															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-22 Receive Buffer Timestamp Register Description

Field	Name	Description
31:16	Reserved	Reserved
15:0	RBRTS	Reception timestamp for CiA 603 time-stamping

29.4.2.18 CAN_RBDATAREG0

0x008c			Receive buffer data0 register											CAN_RBDATAREG0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RBPAYLOADDATA0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RBPAYLOADDATA0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-23 Receive Buffer Data0 Register Description

Field	Name	Description
31:0	RBPAYLOADDATA0	Payload data of the frame

29.4.2.19 CAN_RBDATAREG1

0x0090			Receive buffer data1 register											CAN_RBDATAREG1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RBPAYLOADDATA1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	RBPAYLOADDATA1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-24 Receive Buffer Data1 Register Description

Field	Name	Description
31:0	RBPAYLOADDATA1	Payload data of the frame

29.4.2.20 CAN_TBID

0x0890			Transmit buffer identifier register											CAN_TBID		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	TBTTSEN	Reserved		TBID												
Type	RW	RO		RW												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	TBID															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-25 Transmit Buffer Identifier Register Description

Field	Name	Description
31	TBTTSEN	Transmit timestamp enable For CiA 603 time-stamping the acquisition of a transmit time stamp TTS can be selected: 0: No acquisition of a transmit time stamp for this frame. 1: TTS update enabled This bit is only relevant for frames to be transmitted.
30:29	Reserved	Reserved
28:0	TBID	Frame identifier

29.4.2.21 CAN_TBFORMAT

0x0894			Transmit buffer frame format register											CAN_TBFORMAT		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved										TBSEC	TBRMF	Reserved			TBIDE
Type	RO										RW	RW	RO			RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												TBDLC			
Type	RO												RW			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-26 Transmit Buffer Frame Format Register Description

Field	Name	Description
31:22	Reserved	Reserved
21	TBSEC	Simple or extended content 0: Simple content 1: Extended content
20	TBRMF	Remote Frame 0: Data frame 1: Remote frame
19:17	Reserved	Reserved
16	TBIDE	Identifier extension 0: Standard format: ID(28:18) 1: Extended format: ID(28:0)
15:4	Reserved	Reserved

Field	Name	Description
3:0	TBDLC	Data length code

29.4.2.22 CAN_TBDATAREG0

0x08ac			Transmit buffer data0 register											CAN_TBDATAREG0		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PAYLOAD0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PAYLOAD0															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-27 Transmit Buffer Data0 Register Description

Field	Name	Description
31:0	PAYLOAD0	Payload data of the frame

29.4.2.23 CAN_TBDATAREG1

0x08b0			Transmit buffer data1 register											CAN_TBDATAREG1		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	PAYLOAD1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PAYLOAD1															
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 29-28 Transmit Buffer Data1 Register Description

Field	Name	Description
31:0	PAYLOAD1	Payload data of the frame

Debug (DBG)

This chapter describes the details of the Debug (DBG).

Topics: **Page**

30.1 Introduction

The TPS325M51xx series is built with a STAR processor based on ARMv8-M architecture, which contains extensive debug and trace capabilities. The following components are enabled in CoreSight system:

- Serial Wire/JTAG Debug Port (SWJ-DP)
- Breakpoint Unit (BPU)
- Data Watchpoint Unit (DWT)
- Instrumentation Trace Macrocell (ITM)
- Embedded Trace Macrocell (ETM)
- Trace Port Interface Unit (TPIU)
- System Control Space (SCS)

With these debug components, the visibility of the processor and memory is available. For more information, refer to the following reference manuals:

- Arm China STAR Processor Revision: r1p0 Technical Reference Manual
- Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2
- Arm® CoreSight™ SoC-400 Technical Reference Manual
- Arm® Embedded Trace Macrocell Architecture Specification ETMv4

30.2 Features

The TPS325M51xx series provide the Breakpoint Unit (BPU) which supports eight hardware breakpoint comparators and Data Watchpoint and Trace (DWT) which support four watchpoint comparators. They are used for implementing watchpoints, data tracing, and system profiling.

ETM trace is enabled in TPS325M51xx devices. The ETM is a trace source component that outputs information stream about program execution and data accesses.

ITM can generate timestamp messages in the trace stream and make information on the timing of event generation visible to a trace output port.

TPIU can export trace data from the ETM or the ITM trace source to serial SW pins or parallel trace data pins.

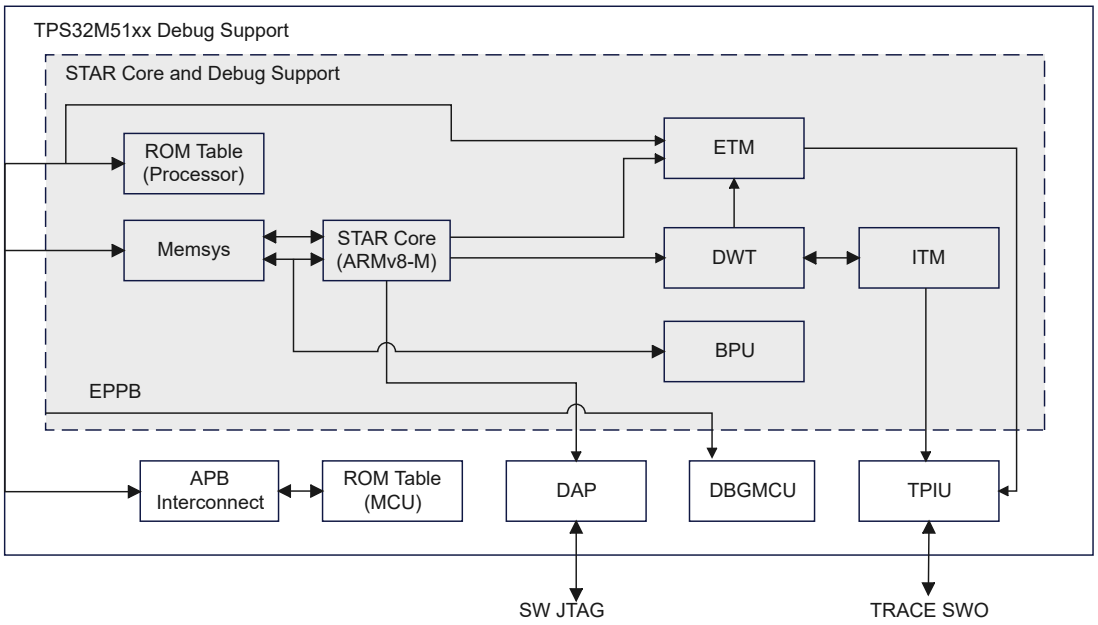


Figure 30-1 TPS325M51xx Debug Support Block Diagram

30.2.1 Breakpoint Unit (BPU)

The Breakpoint Unit (BPU) is configured to provide eight instruction comparators that monitor the instruction fetch address and return a breakpoint instruction when a match is detected.

30.2.2 Data Watchpoint and Trace (DWT)

TPS325M51xx devices implement Data Watchpoint and Trace (DWT), which contains four comparators that can be configured as:

- Hardware watchpoint
- Hardware trace packet for ITM
- ETM triggers
- Instruction address matching
- Data address matching
- Data value matching

The DWT contains counters for:

- Cycles (CYCCNT)
- Folded Instructions (FOLDCNT)
- Additional cycles required to execute all load or store instructions (LSUCNT)
- Sleep cycles (SLEEPCNT)
- Additional cycles required to execute multi-cycle instructions and instruction fetch stalls (CPICNT)
- Cycles spent in exception processing (EXCCNT)

DWT can be configured to generate PC samples at defined intervals, and to generate interrupt event information. And it also provides periodic requests for protocol synchronization to the ITM and the TPIU. DWT is used to trigger watchpoint events caused by a match between the current data or instruction address. Once the match occurs, the comparator generates a watchpoint debug event, on either the PC value or the accessed data address.

30.2.3 Instrumentation Trace Macrocell (ITM)

The Instrumentation Trace Macrocell (ITM) unit, an application-oriented trace source, facilitates printf() style debugging to track both operating system and application events, while producing diagnostic system data. The ITM emits trace information as packets. There are three sources that can generate packets. If multiple sources generate packets simultaneously, the ITM arbitrates the order in which packets are output. The three sources, listed in descending order of priority, are as follows:

1. Software trace

   This involves direct writing to the ITM stimulus registers by the software, which results in packet emission.
2. Hardware trace

   The DWT generates these packets, and the ITM emits them.
3. Time stamping

   Timestamps are emitted relative to packets. TPS325M51xx device contains a 64-bit counter to generate the timestamp automatically just after the trace function is enabled.

The TRCENA bit of the Debug Exception and Monitor Control Register (DEMCR) needs to be enabled before programming or using the ITM.

If the ITM stream requires synchronization packets, it is mandatory to configure the synchronization packet rate in the DWT.

30.2.4 Embedded Trace Macrocell (ETM)

The Embedded Trace Macrocell (ETM) is a trace source component that outputs information stream about program execution and data access. It is configured to instruct tracing only within TPS325M51xx devices. Unlike the ITM, no code instrumentation is needed, and ETM can capture information without affecting the processor’s performance.

The ARM STAR core signals the taken branches and exception events to the ETM trace source, which converts them into trace packets. ETM trace source passes trace packets over the ATB to a trace funnel. This means that all the ATBs can be combined before passing them to the trace sink TPIU.

A debug host should configure the tracing pin at the start, and then ETM data will be available. Otherwise, there is no way to receive ETM data on the ETM bus outside of the MCU. When the trace has been captured, the debugger decompresses it to provide a full disassembly with symbols of the executed code. The debugger links this information with high-level source code and visualizes how the code was executed on the target.

The ETM is useful for profiling code execution and investigating complex software bugs.

30.2.5 Trace Port Interface Unit (TPIU)

The Trace Port Interface Unit (TPIU) is the component that bridges the on-chip trace data from the ETM and the ITM, with separate IDs, to a data stream.

The STAR core embeds a simple TPIU implementation designed for cost-effective debugging, which can be considered as a special version of the CoreSight TPIU.

The TPIU drives trace data to external pins on a target, enabling the Trace Port Analyzer (TPA), typically a component of a debug probe, to capture the trace data. The TPIU's functionalities include:

- Coordinating the stopping of trace capture upon receiving a trigger

- Inserting source identification information into the trace stream, allowing trace data to be re-associated with its trace source
- Outputing the trace data over trace port pins
- Outputing patterns over the trace port

  This pattern output is often referred to as TPIU pattern generation. This allows a TPA to tune its capture logic to the trace port, which maximizes the trace data output frequency on the trace port.

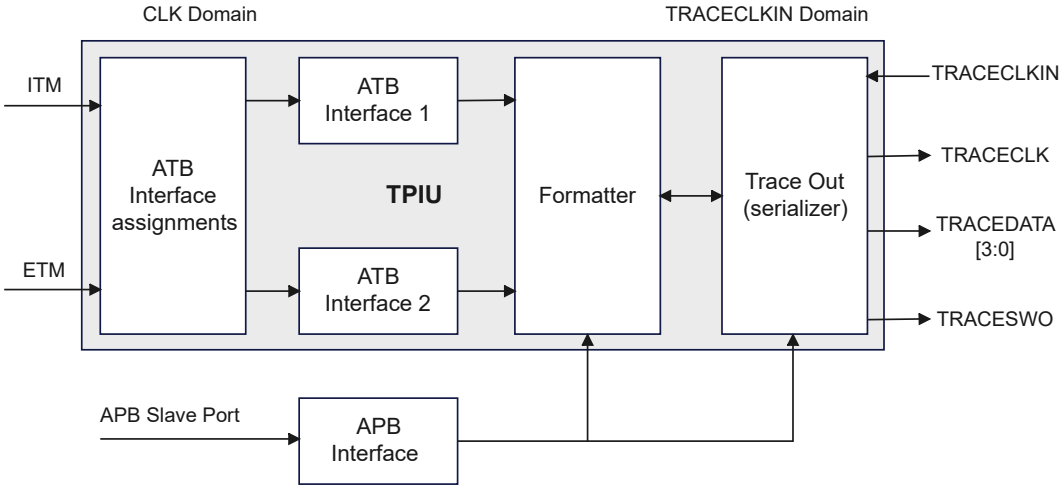


Figure 30-2 Block Diagram of Trace Port Interface Unit (TPIU)

ITM is connected to ATB port 1, and an ETM is connected to ATB port 2, but TPIU does not allow for ITM and ETM trace simultaneously.

The formatter inserts source ID signals into the data packet stream so that trace data can be re-associated with its trace source.

When the formatter is enabled, if there is no data to output after a frame has been started, half-sync packets can be inserted. Distributed synchronization from the DWT or TPIU Periodic Synchronization Control Register (TPIU_PSCR) causes synchronization, which ensures that any partial frame is completed and at least one full synchronization packet is generated.

If the TPIU Selected Pin Protocol Register (TPIU_SPPR) is set to select Trace Port Mode, the formatter is automatically enabled. If one of the SWO modes is selected, the TPIU Formatter and Flush Control Register (TPIU_FFCR) reverts to its previously programmed value.

When SWO mode is selected, TPIU is enabled to bypass the formatter for trace output. If the formatter is bypassed, only the ITM and DWT trace source passes through. The TPIU accepts and discards data from the ETM. This function can be used to connect a device containing an ETM to a trace capture device that is only able to capture SWO data.

30.3 Debug and Trace Port

30.3.1 Debug Port

TPS325M51xx devices integrate Serial Wire/JTAG Debug Port (SWJ-DP) standard CoreSight component which supports two protocols to exchange data with the debug probe:

- The 2-wire Serial Wire Debug (SWD) protocol:
  - SWCLK
  - SWDIO

- The 5-wire JTAG protocol:
  - JTDO
  - JTDI
  - NJTRST
  - JTMS
  - JTCK

NOTE: SWCLK and JTCK, SWDIO and JTMS are multiplexed.

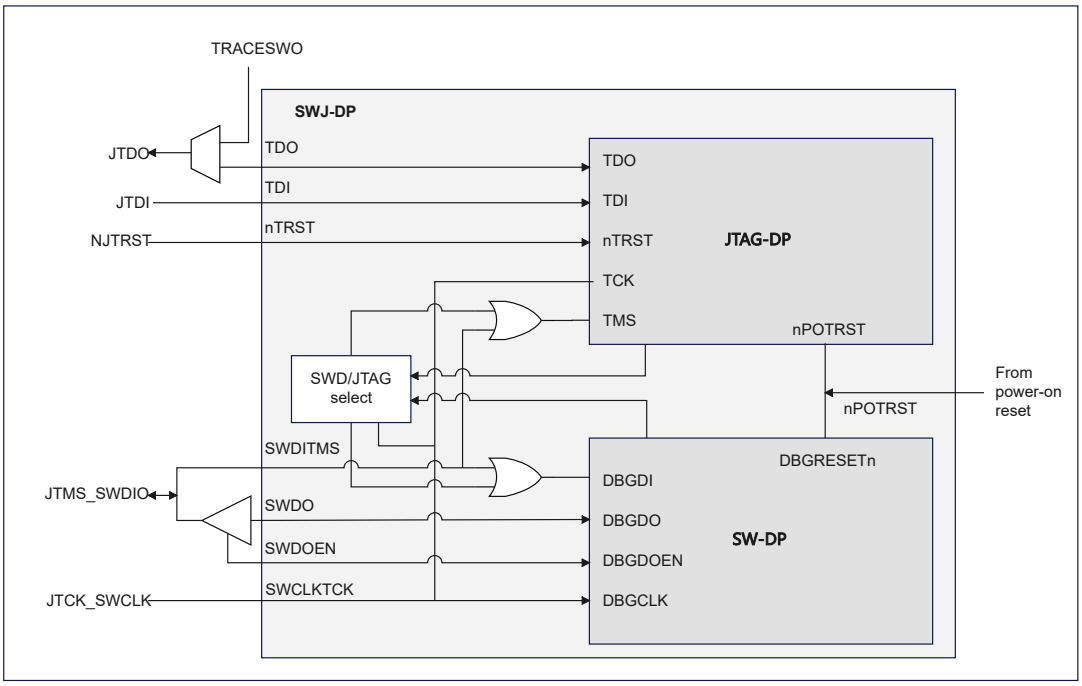


Figure 30-3 SWJ Debug Port

By default, SWJ-DP is on JTAG operation after powerup reset and therefore the JTAG protocol can be used from reset. A switching mechanism allows the switch between SWD and JTAG interfaces.

Switching from JTAG to SWD

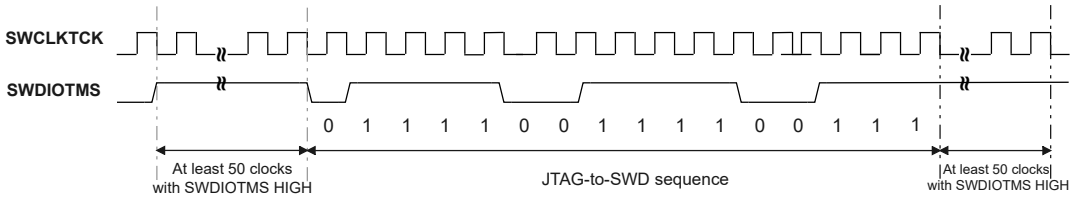


Figure 30-4 JTAG-to-SWD Sequence Timing

1. Send at least 50 SWCLKTCK cycles with SWDIOTMS HIGH. This sequence ensures that the current interface is in its reset state. The JTAG interface only detects the 16-bit JTAG-to-SWD sequence starting from the TLR state.
2. Send the 16-bit JTAG-to-SWD select sequence 0111 1001 1110 0111(MSB first) on SWDIOTMS.
3. Send at least 50 SWCLKTCK cycles with SWDIOTMS HIGH. This sequence ensures that if SWJ-DP was already in SWD operation before sending the select sequence, the SWD interface enters line reset state.

Switching from SWD to JTAG

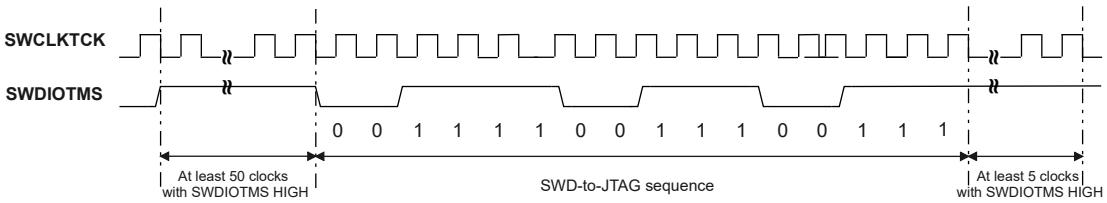


Figure 30-5 SWD-to-JTAG Sequence Timing

1. Send at least 50 SWCLKTCK cycles with SWDIOTMS HIGH.

   This sequence ensures that the current interface is in its reset state. The SWD interface only detects the 16-bit SWD-to-JTAG sequence when it is in the reset state.
2. Send the 16-bit SWD-to-JTAG select sequence 0b0011 1100 1110 0111(MSB first) on SWDIOTMS.
3. Send at least five SWCLKTCK cycles with SWDIOTMS HIGH.

   This sequence ensures that if SWJ-DP was already in JTAG operation before sending the select sequence, the JTAG TAP enters the Test-Logic-Reset state.

30.3.2 Trace Port

In relation to the trace output, the TPIU provides two options:

- The asynchronous 1-wire trace port, known as Serial Wire Output (SWO).
- The synchronous 5-wire trace port, comprising a clock and 1, 2 or 4 bits of data: TRACECK, TRACED[3:0].

NOTE: SWO pin is multiplex with multiplexed with the JTAG JTDO signal so that it cannot be used at the same time as the JTAG protocol. SWD must be chosen. It requires a constant frequency for TRACECLKIN.

For the standard UART (NRZ) capture mechanism, 5% accuracy is needed. The Manchester encoded version is tolerant up to 10%.The trace pins are used in JTAG mode and TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used. It is not required to provide a stable clock frequency.

The TRACE pins (including TRACECK) are driven by the rising edge of TRACLKIN (equal to HCLK). Consequently, the output frequency of TRACECK is equal to HCLK/2.SWO pin and trace pins cannot be used at the same time.

30.4 Pinout of Debug and Trace Port

30.4.1 Debug and Trace Pins Assignment

Debug and trace pins assignment shown in the table below are available for all TPS325M51xx devices.

Table 30-1 Pin Assignment of JTAG/SWD and Trace Port

Pin Name	Pin Assignment	Direction	Description
TRACED3	PD8	O	Trace data pin 3
TRACED2	PD11	O	Trace data pin 2

Pin Name	Pin Assignment	Direction	Description
TRACED1	PD12	O	Trace data pin 1
TRACED0	PE0	O	Trace data pin 0
TRACECK	PE1	O	Trace clock
NJTRST	PE2	I	JTAG Test nReset
JTDO_TRACESWO	PE3	O	JTDO: JTAG Test Data Output TRACESWO: Serial Wire Output Trace
JTDI	PE4	I	JTAG Test Data Input
JTMS_SWDIO	PE5	I/O	JTMS: JTAG Test Mode Selection SWDIO: Serial Wire Data Input/ Output
JTCK_SWCLK	PE6	I	JTCK: JTAG Test Clock SWCLK: Serial Wire Clock

30.4.1.1 Debug Pins Assignment

After reset, all debug pins are assigned for immediate usage.

TPS325M51xx MCUs also offer the possibility of disabling some or all of the debug pins and release the pins for General-Purpose I/O (GPIO) usage.

NJTRST can be left disconnected but cannot be used as GPIO when a debugger is connecting.

30.4.1.2 Trace Pins Assignment

After reset, all trace pins are at a low level, and the debug host should be configured for trace usage. Notice that 1 wire SWO trace pin is only available if using serial wire mode (not JTAG mode), while ETM trace data pins are available in JTAG mode (not in SWD mode).

30.4.2 Internal Pull-up and Pull-down on Debug Pins

During and after reset, the multiplexing function is not active; most general-purpose I/O, including debug pins, are under AF mode to avoid any uncontrolled IO levels. Details can be found as below:

- JTDI: Pull-up
- JTCK_SWCLK: Pull-down
- JTMS_SWDIO: Pull-up
- NJTRST: Pull-up
- JTDO_TRACESWO: Floating state no pull-up/pull-down

30.5 MCU Debug Component (DBGMCU)

The MCU debug component helps the debugger support for:

- Low-power modes
- Clock control for timers, watchdog, and I2C during a breakpoint

30.5.1 Debug Support for Low-Power Mode

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow HCLK to be turned off during a debug session. As the clock is required for the debugger connection during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In sleep mode, the DBGSLEEP bit of the DBGMCU_CR register must be previously set by the debugger. This feeds the HCLK with the same clock that is provided to the system clock.
- In stop mode, the DBGSTOP bit must be previously set by the debugger. This enables the internal RC oscillator clock to feed HCLK in stop mode.
- In standby mode, the DBGSTANDBY bit must be previously set by the debugger. This keeps the regulators on and enables the internal RC oscillator clock to feed HCLK in standby mode. A system reset is generated internally so that exiting from standby is identical than fetching from reset.
- In shutdown mode, the DBGSHUTDOWN bit must be previously set by the debugger. This keeps the regulators on and enables the internal RC oscillator clock to feed HCLK in Shutdown mode. A system reset is generated internally so that exiting from shutdown is identical than fetching from reset.

The DBGMCU_APBFZR register can be written by the debugger under system reset. If the debugger host does not support these features, it is still possible to write this register by software.

30.5.2 Debug Support for Timers, RTC, Watchdog and I2C

During a breakpoint, it is necessary to choose how the counter of timers, RTC and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes. For the I2C, the user can choose to block the SMBus timeout during a breakpoint.

The DBGMCU_APBFZR registers can be written by the debugger under system reset. If the debugger host does not support these features, it is still possible to write these registers by software.

30.6 Registers

30.6.1 Register Address Map

Offset	Register Name	Register Description
0x0000	DBGMCU_CR	Debug MCU control register
0x0004	DBGMCU_APBFZR	Debug MCU APB freeze register

30.6.2 Register Field Details

30.6.2.1 DBGMCU_CR

0x0000			Debug MCU control register											DBGMCU_CR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved															
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved												DBGSTOP	DBGSHUTDOWN	DBGSTANDBY	DBGSLEEP
Type	RO												RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-2 Debug MCU Control Register Description

Field	Name	Description
31:4	Reserved	Reserved

Field	Name	Description
3	DBGSTOP	Debug stop mode 0: HCLK = Off In stop mode, the clock controller disables HCLK. When exiting from stop mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI16)). Consequently, the software must reprogram the clock controller to enable the PLL, the XTAL and so on. 1: HCLK = On In this case, when entering stop mode, HCLK is provided by the internal RC oscillator, which remains active in stop mode. When exiting stop mode, it is necessary for the software to reprogram the clock controller to enable the PLL, the XTA, and so on, similar to what would be done if DBGSTOP = 0.
2	DBGSHUTDOWN	Debug shutdown mode 0: HCLK = Off 1: HCLK = On In this case, the digital part is not unpowered, and HCLK is provided by the internal RC oscillator, which remains active. In addition, MCU generates a system reset during standby mode so that exiting from shutdown is identical to fetching from reset.
1	DBGSTANDBY	Debug standby mode 0: HCLK = Off The whole digital part is unpowered. From a software perspective, exiting standby mode is almost the same as retrieving the reset vector, with the only difference being a few status bits that indicate the MCU is resuming from standby mode. 1: HCLK = On In this case, the digital part is not unpowered, and HCLK is provided by the internal RC oscillator, which remains active. In addition, MCU generates a system reset during standby mode so that exiting from standby is identical to fetching from reset.

Field	Name	Description
0	DBGSLEEP	Debug sleep mode 0: HCLK = Off In sleep mode, HCLK is disabled. In sleep mode, the clock controller configuration is not reset and remains previously programmed. Consequently, the software does not need to reconfigure the clock controller when exiting from sleep mode. 1: HCLK = On In this case, when entering sleep mode, HCLK is fed by the same clock provided to the system clock previously configured by the software.

30.6.2.2 DBGMCU_APBFZR

0x0004			Debug MCU APB freeze register											DBGMCU_APBFZR		
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Reserved									DBGI2C1STOP	DBGI2C0STOP	Reserved				
Type	RO									RW	RW	RO				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved			DBGIWDGSTOP	DBGWWDGSTOP	DBGRTCSTOP	Reserved	DBGADVTMR2STOP	DBGADVTMR1STOP	DBGADVTMR0STOP	DBGGPTMR5STOP	DBGGPTMR4STOP	DBGGPTMR3STOP	DBGGPTMR2STOP	DBGGPTMR1STOP	DBGGPTMR0STOP
Type	RO			RW	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 30-3 Debug MCU APB Freeze Register Description

Field	Name	Description
31:23	Reserved	Reserved
22	DBGI2C1STOP	I2C1 SMBUS timeout counter stopped when core is halted 0: Same behavior as in normal mode 1: The I2C1 SMBus timeout is frozen
21	DBGI2C0STOP	I2C0 SMBUS timeout counter stopped when core is halted 0: Same behavior as in normal mode 1: The I2C0 SMBus timeout is frozen
20:13	Reserved	Reserved
12	DBGIWDGSTOP	Independent Watchdog counter stopped when core is halted 0: The Independent Watchdog counter clock continues even if the core is halted 1: The Independent Watchdog counter clock is stopped when the core is halted

Field	Name	Description
11	DBGWWDGSTOP	Window Watchdog counter stopped when core is halted 0: The Window Watchdog counter clock continues even if the core is halted 1: The Window Watchdog counter clock is stopped when the core is halted
10	DBGRTCSTOP	RTC counter stopped when core is halted 0: The clock of the RTC counter is fed even if the core is halted 1: The clock of the RTC counter is stopped when the core is halted
9	Reserved	Reserved
8	DBGADVTMR2STOP	ADVTMR2 counter stopped when core is halted 0: The counter clock of ADVTMR2 is fed even if the core is halted 1: The counter clock of ADVTMR2 is stopped when the core is halted
7	DBGADVTMR1STOP	ADVTMR1 counter stopped when core is halted 0: The counter clock of ADVTMR1 is fed even if the core is halted 1: The counter clock of ADVTMR1 is stopped when the core is halted
6	DBGADVTMR0STOP	ADVTMR0 counter stopped when core is halted 0: The counter clock of ADVTMR0 is fed even if the core is halted 1: The counter clock of ADVTMR0 is stopped when the core is halted
5	DBGGPTMR5STOP	GPTMR5 counter stopped when core is halted 0: The counter clock of GPTMR5 is fed even if the core is halted 1: The counter clock of GPTMR5 is stopped when the core is halted
4	DBGGPTMR4STOP	GPTMR4 counter stopped when core is halted 0: The counter clock of GPTMR4 is fed even if the core is halted 1: The counter clock of GPTMR4 is stopped when the core is halted
3	DBGGPTMR3STOP	GPTMR3 counter stopped when core is halted 0: The counter clock of GPTMR3 is fed even if the core is halted 1: The counter clock of GPTMR3 is stopped when the core is halted

Field	Name	Description
2	DBGGPTMR2STOP	GPTMR2 counter stopped when core is halted 0: The counter clock of GPTMR2 is fed even if the core is halted 1: The counter clock of GPTMR2 is stopped when the core is halted
1	DBGGPTMR1STOP	GPTMR1 counter stopped when core is halted 0: The counter clock of GPTMR1 is fed even if the core is halted 1: The counter clock of GPTMR1 is stopped when the core is halted
0	DBGGPTMR0STOP	GPTMR0 counter stopped when core is halted 0: The counter clock of GPTMR0 is fed even if the core is halted 1: The counter clock of GPTMR0 is stopped when the core is halted

Revision History

Page

- Initial released.....

IMPORTANT NOTICE AND DISCLAIMER

Trademarks. Any of the 思瑞浦 or 3PEAK trade names, trademarks, graphic marks, and domain names contained in this document /material are the property of 3PEAK. You may NOT reproduce, modify, publish, transmit or distribute any Trademark without the prior written consent of 3PEAK.

Performance Information. Performance tests or performance range contained in this document/material are either results of design simulation or actual tests conducted under designated testing environment. Any variation in testing environment or simulation environment, including but not limited to testing method, testing process or testing temperature, may affect actual performance of the product.

Disclaimer. 3PEAK provides technical and reliability data (including data sheets), design resources (including reference designs), application or other design recommendations, networking tools, security information and other resources "As Is". 3PEAK makes no warranty as to the absence of defects, and makes no warranties of any kind, express or implied, including without limitation, implied warranties as to merchantability, fitness for a particular purpose or non-infringement of any third-party's intellectual property rights. Unless otherwise specified in writing, products supplied by 3PEAK are not designed to be used in any life-threatening scenarios, including critical medical applications, automotive safety-critical systems, aviation, aerospace, or any situations where failure could result in bodily harm, loss of life, or significant property damage. 3PEAK disclaims all liability for any such unauthorized use.