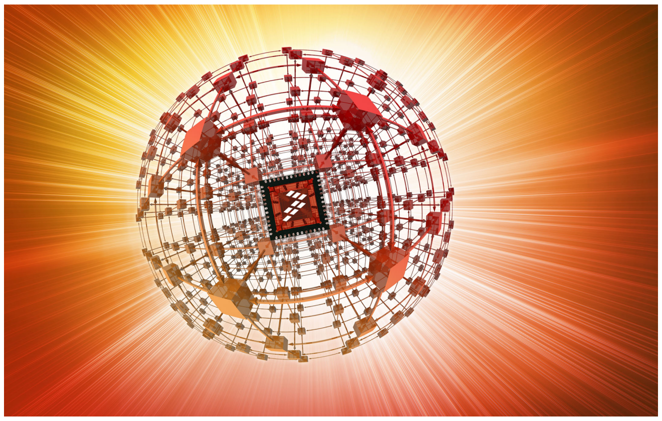
KE04 Sub-Family Reference Manual

Supports: MKE04Z8VTG4(R), MKE04Z8VWJ4(R) and MKE04Z8VFK4(R)



Document Number: MKE04P24M48SF0RM
Rev 3, Feburary 2014

Contents

Section number **Title** **Page**

Section number	Title	Page

Section number Title Page

Section number Title Page

Section number Title Page

Section number | Title | Page

Section number	Title	Page

Section number	Title	Page

Section number	Title	Page

Section number	Title	Page

Section number	Title	Page

Section number	Title	Page

Section number	Title	Page

Section number Title Page

Chapter 1
About This Document

1.1 Overview

1.1.1 Purpose

This document describes the features, architecture, and programming model of the Freescale KE04 microcontroller.

1.1.2 Audience

This document is primarily for system architects and software application developers who are using or considering using the KE04 microcontroller in a system.

1.2 Conventions

1.2.1 Numbering systems

The following suffixes identify different numbering systems:

This suffix	Identifies a
b	Binary number. For example, the binary equivalent of the number 5 is written 101b. In some cases, binary numbers are shown with the prefix *0b*.
d	Decimal number. Decimal numbers are followed by this suffix only when the possibility of confusion exists. In general, decimal numbers are shown without a suffix.
h	Hexadecimal number. For example, the hexadecimal equivalent of the number 60 is written 3Ch. In some cases, hexadecimal numbers are shown with the prefix *0x*.

1.2.2 Typographic notation

The following typographic notation is used throughout this document:

Example	Description
placeholder, x	Items in italics are placeholders for information that you provide. Italicized text is also used for the titles of publications and for emphasis. Plain lowercase letters are also used as placeholders for single letters and numbers.
`code`	Fixed-width type indicates text that must be typed exactly as shown. It is used for instruction mnemonics, directives, symbols, subcommands, parameters, and operators. Fixed-width type is also used for example code. Instruction mnemonics and directives in text and tables are shown in all caps; for example, BSR.
SR[SCM]	A mnemonic in brackets represents a named field in a register. This example refers to the Scaling Mode (SCM) field in the Status Register (SR).
REVNO[6:4], XAD[7:0]	Numbers in brackets and separated by a colon represent either: • A subset of a register's named field For example, REVNO[6:4] refers to bits 6–4 that are part of the COREREV field that occupies bits 6–0 of the REVNO register. • A continuous range of individual signals of a bus For example, XAD[7:0] refers to signals 7–0 of the XAD bus.

1.2.3 Special terms

The following terms have special meanings:

Term	Meaning
asserted	Refers to the state of a signal as follows: • An active-high signal is asserted when high (1). • An active-low signal is asserted when low (0).
deasserted	Refers to the state of a signal as follows: • An active-high signal is deasserted when low (0). • An active-low signal is deasserted when high (1). In some cases, deasserted signals are described as *negated*.
reserved	Refers to a memory space, register, or field that is either reserved for future use or for which, when written to, the module or chip behavior is unpredictable.

Chapter 2 Introduction

2.1 Overview

This chapter provides an overview of the Kinetis KE04 product family of ARM® Cortex®-M0+ MCUs. It also presents high-level descriptions of the modules available on the devices covered by this document.

2.2 Module functional categories

The modules on this device are grouped into functional categories. The following sections describe the modules assigned to each category in more detail.

Table 2-1. Module functional categories

Module category	Description
ARM Cortex-M0+ core	• 32-bit MCU core from ARM's Cortex-M class, 1.77 CoreMark®/MHz from single-cycle access memories, 48 MHz CPU frequency
System	• System integration module (SIM) • Power management and mode controllers (PMC) • Miscellaneous control module (MCM) • Bit manipulation engine (BME) • Peripheral bridge (AIPS) • Watchdog (WDOG)
Memories	• Internal memories include: • Up to 8 KB flash memory • Up to 1 KB SRAM
Clocks	• External crystal oscillator or resonator • Low range: 31.25–39.0625 kHz • High range: 4– 24 MHz • External square wave input clock • Internal clock references • 31.25 to 39.0625 kHz oscillator • 1 kHz LPO oscillator • Frequency-locked loop (FLL) range: 40–50 MHz

Table continues on the next page...

Table 2-1. Module functional categories (continued)

Module category	Description
Security	• Watchdog (WDOG) with independent clock source • Cyclic redundancy check (CRC) module for error detection
Analog	• One 12-bit analog-to-digital converters (ADC) with up to 12 channels • Two analog comparators (ACMP) with internal 6-bit digital-to-analog converter (DAC)
Timers	• One 6-channel FlexTimer (FTM) with full function • One 2-channel FTM with basic TPM function • 2-channel periodic interrupt timer (PIT) • Real time clock (RTC) • One pulse width timer (PWT) • System tick timer (SysTick)
Communications	• One 8-bit serial peripheral interface (SPI) • One inter-integrated circuit (I^2C) module • One universal asynchronous receiver/transmitter (UART) module
Human-Machine Interfaces (HMI)	• General purpose input/output (GPIO) controller • Two keyboard Interrupt (KBI) • Interrupt (IRQ)

2.2.1 ARM Cortex-M0+ core modules

The following core modules are available on this device.

Table 2-2. Core modules

Module	Description
ARM Cortex-M0+	The ARM Cortex-M0+ is the newest member of the Cortex M Series of processors targeting microcontroller applications focused on very cost sensitive, deterministic, interrupt driven environments. The Cortex M0+ processor is based on the ARMv6 Architecture and Thumb®-2 ISA and is 100% instruction set compatible with its predecessor, the Cortex-M0 core, and upward compatible to Cortex-M3 and M4 cores. The ARM Cortex-M0+ improvements include an ARMv6 Thumb-2 DSP, ported from the ARMv6-A/R profile architectures, that provide 32-bit instructions with SIMD (single instruction multiple data) DSP style multiply-accumulates and saturating arithmetic to support single cycle 32x32 multiplier.
Nested vectored interrupt controller (NVIC)	The ARMv6-M exception model and nested-vectored interrupt controller (NVIC) implement a relocatable vector table supporting many external interrupts, a single non-maskable interrupt (NMI), and priority levels. The NVIC replaces shadow registers with equivalent system and simplified programmability. The NVIC contains the address of the function to execute for a particular handler. The address is fetched via the instruction port allowing parallel register stacking and look-up. The first sixteen entries are allocated to ARM internal sources with the others mapping to MCU-defined interrupts.
Asynchronous wakeup interrupt controller (AWIC)	The primary function of the Asynchronous Wake-up Interrupt Controller (AWIC) is to detect asynchronous wake-up events in stop modes and signal to clock control logic to resume system clocking. After clock restart, the NVIC observes the pending interrupt and performs the normal interrupt or event processing.

Table continues on the next page...

Table 2-2. Core modules (continued)

Module	Description
Single-cycle I/O port (IOPORT)	For high-speed, single-cycle access to peripherals, the Cortex-M0+ processor implements a dedicated single-cycle I/O port. On this device, fast GPIO (FGPIO) is implemented on IOPORT interface.
Debug interfaces	Most of this device's debug is based on the ARM CoreSight™ architecture. One debug interface is supported: • Serial Wire Debug (SWD)

2.2.2 System modules

The following system modules are available on this device.

Table 2-3. System modules

Module	Description
System integration module (SIM)	The SIM includes integration logic and several module configuration settings.
Power management controller (PMC)	The PMC provides the user with multiple power options. Multiple modes are supported that allow the user to optimize power consumption for the level of functionality needed. Includes power-on-reset (POR) and integrated low voltage detect (LVD) with reset (brownout) capability and selectable LVD trip points.
Miscellaneous control module (MCM)	The MCM includes integration logic and details.
Peripheral bridge (AIPS-Lite)	The peripheral bridge converts the ARM AHB interface to an interface to access a majority of peripherals on the device.
Watchdog (WDOG)	The WDOG monitors internal system operation and forces a reset in case of failure. It can run from an independent 1 kHz low-power oscillator with a programmable refresh window to detect deviations in program flow or system frequency.
Bit manipulation engine (BME)	The BME provides hardware support for atomic read-modify-write memory operations to the peripheral address space in Cortex-M0+ based microcontrollers.

2.2.3 Memories and memory interfaces

The following memories and memory interfaces are available on this device.

Table 2-4. Memories and memory interfaces

Module	Description
Flash memory (FTMRE)	Flash memory — up to 8 KB of the non-volatile flash memory that can execute program code.
Flash memory controller (FMC)	FMC is a memory acceleration unit that provides an interface between Cortex M0+ core and the 32-bit program flash memory. The FMC contains one 32-bit

Table continues on the next page...

Table 2-4. Memories and memory interfaces (continued)

Module	Description
	speculation buffer and one 32-byte cache that can accelerate instruction fetch and flash access.
SRAM	Up to 1 KB internal system RAM, supporting bit operation through BME module or aliased bit-band domain.

2.2.4 Clocks

The following clock modules are available on this device.

Table 2-5. Clock modules

Module	Description
Internal Clock Source (ICS)	ICS module containing an internal reference clock (ICSIRCLK) and a frequency-locked-loop (FLL).
System oscillator (OSC)	The system oscillator, in conjunction with an external crystal or resonator, generates a reference clock for the MCU.
Low-Power Oscillator (LPO)	The PMC module contains a 1 kHz low-power oscillator which acts as a standalone low-frequency clock source in all modes.

2.2.5 Security and integrity modules

The following security and integrity modules are available on this device:

Table 2-6. Security and integrity modules

Module	Description
Cyclic Redundancy Check (CRC)	CRC generates 16/32-bit CRC code for error detection.
Watchdog (WDOG)	The WDOG monitors internal system operation and forces a reset in case of failure. It can run from an independent 1 kHz low-power oscillator with a programmable refresh window to detect deviations in program flow or system frequency.

2.2.6 Analog modules

The following analog modules are available on this device:

Table 2-7. Analog modules

Module	Description
Analog-to-digital converters (ADC)	12-bit successive-approximation ADC module with up to 12 channels.
Analog comparators (ACMP)	Two comparators with support of analog input voltages across the full range of the supply voltage and CPU interrupt. ACMP0/1 is further capable to trigger an ADC acquisition and FTM update.
6-bit digital-to-analog converters (DAC)	64-tap resistor ladder network which provides a selectable voltage reference for comparator.

2.2.7 Timer modules

The following timer modules are available on this device:

Table 2-8. Timer modules

Module	Description
FlexTimer modules (FTM)	• Selectable FTM source clock, supporting separate timer clock up to 48 MHz, programmable prescaler • 16-bit counter supporting free-running or initial/final value, and counting is up or up-down • Input capture, output compare, and edge-aligned and center-aligned PWM modes • Operation of FTM channels as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs • Deadtime insertion is available for each complementary pair • Software control of PWM outputs • Up to two fault inputs for global fault control • Configurable channel polarity • Programmable interrupt on input capture, reference compare, overflowed counter, or detected fault condition
Periodic interrupt timers (PIT)	• One general purpose interrupt timer • Interrupt timers for triggering ADC conversions • 32-bit counter resolution • Clocked by bus clock frequency
Real-time counter (RTC)	• 16-bit up-counter • 16-bit modulo match limit • Software controllable periodic interrupt on match • Software selectable clock sources for input to prescaler with programmable 16-bit prescaler • Bus clock • IRC clock (31.25~39.0625 kHz) • LPO (~1 kHz) • System oscilator output clock
Pulse Width Timer (PWT)	• Automatic measurement of pulse width with 16-bit resolution • Separate positive and negative pulse width measurements • Programmable triggering edge for starting measurement • Programmable measuring time between successive alternating edges, rising edges or falling edges • Programmable pre-scaler from clock input as 16-bit counter time base • Two selectable clock sources, supporting separate timer clock up to 48 MHz

Table 2-8. Timer modules

Module	Description
	• Four selectable pulse inputs • Programmable interrupt generation upon pulse width value updated and counter overflow

2.2.8 Communication interfaces

The following communication interfaces are available on this device:

Table 2-9. Communication modules

Module	Description
Serial peripheral interface (SPI)	One SPI synchronous serial bus for communication to an external device.
Inter-integrated circuit (I2C)	One I2C module for inter device communications. Also supports the System Management Bus (SMBus) Specification, version 2.
Universal asynchronous receiver/ transmitters (UART)	One asynchronous serial bus communication interface (UART) modules with optional 13-bit break, full duplex non-return to zero (NRZ), and LIN extension support.

2.2.9 Human-machine interfaces

The following human-machine interfaces (HMI) are available on this device:

Table 2-10. HMI modules

Module	Description
General purpose input/output (GPIO)	Up to 22 general purpose input or output (GPIO) pins.
Keyboard Interrupts (KBI)	Two KBI modules to support pin interrupts.
Interrupt (IRQ)	IRQ module provides a maskable interrupt input.

2.2.10 Orderable part numbers

The following table summarizes the part numbers of the devices covered by this document.

Table 2-11. Orderable part numbers summary

Freescale part number	CPU frequency	Pin count	Package	Total flash memory	RAM	Temperature range
MKE04Z8VTG4(R)	48 MHz	16	TSSOP	8 KB	1 KB	-40 to 105 °C
MKE04Z8VWJ4(R)	48 MHz	20	SOIC	8 KB	1 KB	-40 to 105 °C
MKE04Z8VFK4(R)	48 MHz	24	QFN	8 KB	1 KB	-40 to 105 °C

Chapter 3
Chip Configuration

3.1 Introduction

This chapter provides details on the individual modules of the microcontroller. It includes:

- Module block diagrams showing immediate connections within the device
- Specific module-to-module interactions not necessarily discussed in the individual module chapters
- Links for more information

3.2 Module to Module Interconnects

3.2.1 Interconnection overview

The following table captures the module-to-module interconnections for this device.

Table 3-1. Module-to-module interconnects

Peripheral	Signal	—	to peripheral	Use case	Control	Comment
FTM2	INITTRG, MatchTRG	to	ADC (Trigger)	ADC Triggering	SIM_SOPT[ADHWT]	—
PIT	TIF0,TIF1	to	ADC (Trigger)	ADC Triggering	SIM_SOPT[ADHWT]	—
RTC	RTC Overflow	to	ADC (Trigger)	ADC Triggering	SIM_SOPT[ADHWT]	—
FTM0	INITTRG	to	ADC (Trigger)	ADC Triggering	SIM_SOPT[ADHWT]	—
ACMP0	CMP0_OUT	to	ADC (Trigger)	ADC Triggering	SIM_SOPT[ADHWT]	—
ACMP1	CMP1_OUT	to	ADC (Trigger)	ADC Triggering	SIM_SOPT[ADHWT]	—
RTC	RTC Overflow	to	FTM0_CH0	FTM0 capture input	SIM_SOPT[FTMIC]	—
ACMP0	CMP0_OUT	to	FTM0_CH0	FTM0 capture input	SIM_SOPT[FTMIC]	—
ACMP1	CMP1_OUT	to	FTM0_CH0	FTM0 capture input	SIM_SOPT[FTMIC]	—

Table continues on the next page...

Table 3-1. Module-to-module interconnects (continued)

Peripheral	Signal	—	to peripheral	Use case	Control	Comment
ACMP0	CMP0_OUT	to	UART0_RxD	UART0_RxD filter	SIM_SOPT[RXDFE]	—
ACMP1	CMP1_OUT	to	UART0_RxD	UART0_RxD filter	SIM_SOPT[RXDFE]	—
UART0	UART0_TxD	to	Modulated by FTM0 CH0	UART modulation	SIM_SOPT[TXDME]	—
UART0	UART0_RxD	to	Tagged by FTM0 CH1	UART modulation	SIM_SOPT[RXDCE]	—
ACMP0	CMP0_OUT	to	FTM2 Trigger0	FTM2 Trigger input	SIM_SOPT[ACTRG] and FTM2_SYNC[TRIG0]	—
ACMP1	CMP1_OUT	to	FTM2 Trigger0	FTM2 Trigger input	SIM_SOPT[ACTRG] and FTM2_SYNC[TRIG0]	—
FTM0	FTM0 CH0 Output	to	FTM2 Trigger1	FTM2 Trigger input	FTM2_SYNC[TRIG1]	—
FTM2	FTMSYNC	to	FTM2 Trigger2	FTM2 Trigger input	SIM_SOPT[FTMSYNC] and FTM2_SYNC[TRIG2]	—
ACMP0	CMP0_OUT	to	FTM2 Fault0	FTM2 fault input	FTM2_FLTCTRL[FAULT0EN]	—
EXTRG_IN	EXTRG_IN	to	FTM2 Fault1	FTM2 fault input	FTM2_FLTCTRL[FAULT1EN]	PTA6
EXTRG_IN	EXTRG_IN	to	FTM2 Fault2	FTM2 fault input	FTM2_FLTCTRL[FAULT2EN]	PTA7
ACMP1	CMP1_OUT	to	FTM2 Fault3	FTM2 fault input	FTM2_FLTCTRL[FAULT3EN]	—
EXTRG_IN	EXTRG_IN	to	PWT	PWT_IN0	PWT_R1[PINSEL]	PTC4
EXTRG_IN	EXTRG_IN	to	PWT	PWT_IN1	PWT_R1[PINSEL]	PTB0
ACMP0	CMP0_OUT	to	PWT	PWT_IN2	PWT_R1[PINSEL]	
ACMP1	CMP1_OUT	to	PWT	PWT_IN3	PWT_R1[PINSEL]	

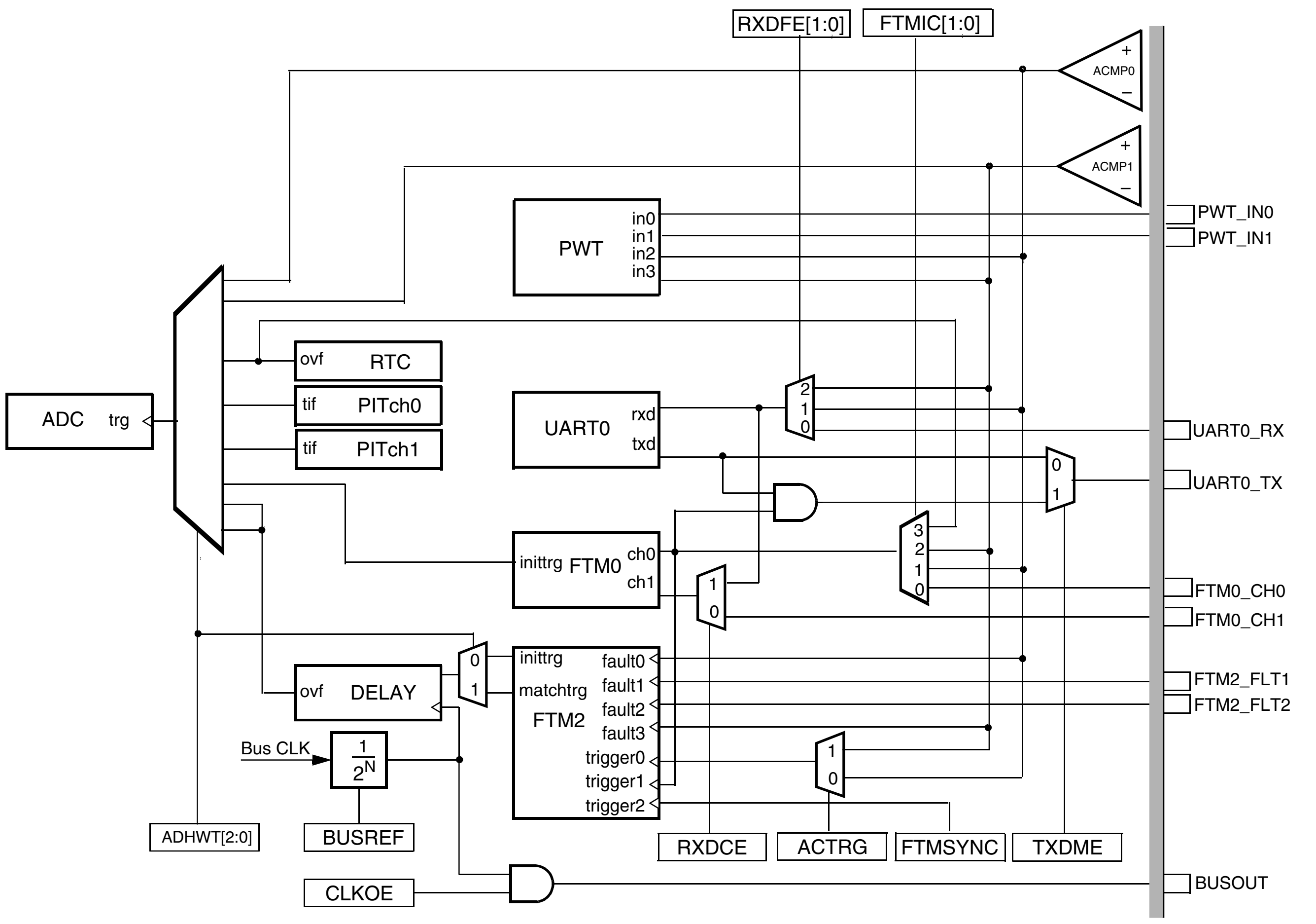


Figure 3-1. System interconnection diagram

3.2.2 Analog reference options

Several analog blocks have selectable reference voltages as shown in Table 3-2. These options allow analog peripherals to share or have separate analog references. Care must be taken when selecting analog references to avoid cross talk noise.

Table 3-2. Analog reference options

Module	Reference option	Comment/ reference selection
12-bit ADC	• VREFH • VREFL	VREFH is internally connected to VDD, VREFL is internally connected to VSS.
ACMP0 ACMP1	• VDDA • Bandgap	Selected by ACMP0_C1[DACREF] or ACMP1_C1[DACREF]

3.2.3 ACMP output capture

When SIM_SOPT[RXDFE] is set, one of the ACMP outputs can be selected to work as a filtered receiver channel of UART0. When SIM_SOPT[FTM IC] is set, one of the ACMP outputs can be selected to be captured by FTM0_CH0. When SIM_SOPT[ACTRG] is set, one of the ACMP outputs can be selected to work as FTM2 trigger input.

3.2.4 UART0_TX modulation

UART0_TX can be modulated by FTM0 channel 0 output. When SIM_SOPT[TXDME] is set, the UART0_TX is gated by FTM0 channel0 output through an AND gate, and then mapped to UART0_TX pinout. When this field is clear, the UART0_TX is directly mapped on the pinout. To enable IR modulation function, both FTM0_CH0 and UART must be active. The FTM0_CNT and FTM0_MOD registers specify the period of the PWM, and the FTM0_C0V register specifies the duty cycle of the PWM. Then, when SIM_SOPT[TXDME] is enabled, each data transmitted via UART0_TX from UART0 is modulated by the FTM0 channel 0 output, and the FTM0_CH0 pin is released to other shared functions regardless of the configuration of FTM0 pin reassignment.

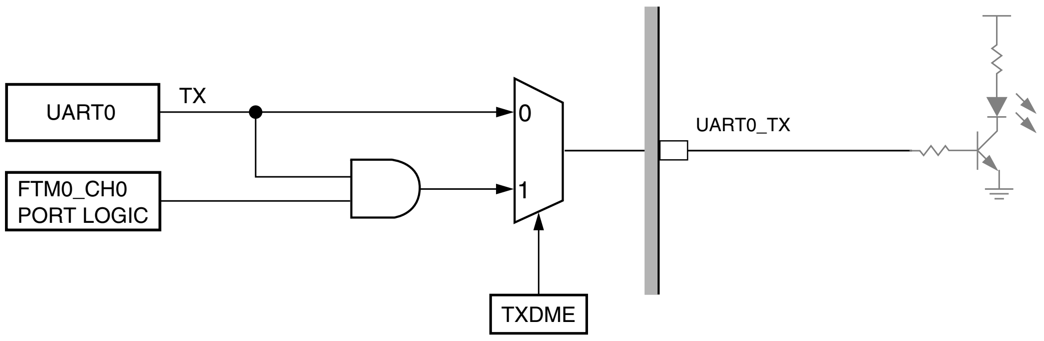


Figure 3-2. IR modulation diagram

3.2.5 UART0_RX capture

UART0_RX pin is selectable connected to UART0 module directly or tagged to FTM0 channel 1. When SIM_SOPT[RXDCE] is set, the UART0_RX pin is connected to both UART0 and FTM0 channel 1, and the FTM0_CH1 pin is released to other shared functions regardless of the configuration of FTM0 pin reassignment. When this field is clear, the UART0_RX pin is connected to UART0 only.

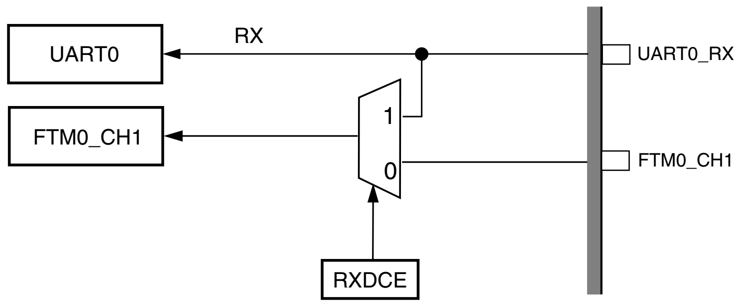


Figure 3-3. UART0_RX capture function diagram

3.2.6 UART0_RX filter

When SIM_SOPT[RXDFE] is clear, the UART0_RX pin is connected to UART0 module directly. When this field is correctly set, the ACMP output can be connected to the receive channel of UART0. To enable UART0_RX filter function, both UART0 and ACMP must be active. If this function is active, the UART0 external UART0_RX pin is released to other shared functions regardless of the configuration of UART0 pin reassignment. When UART0_RX capture function is active, the ACMP output is injected to FTM0 channel 1 as well.

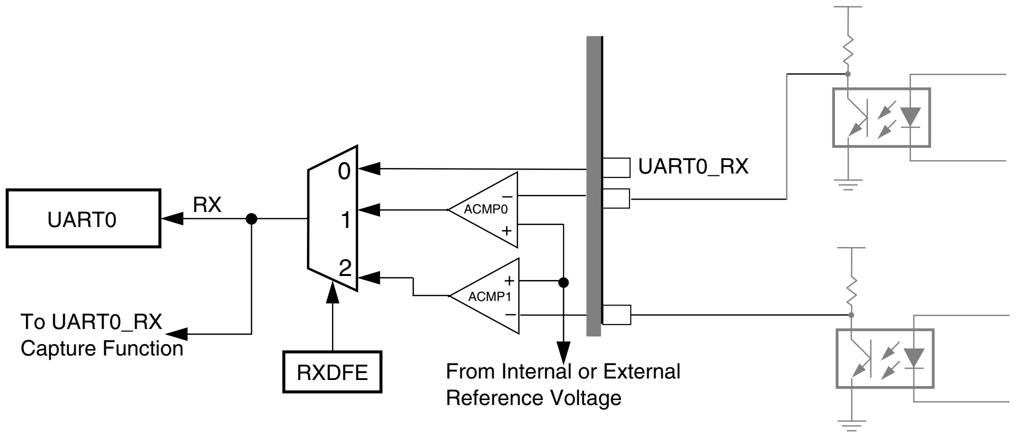

UART0
RX
To UART0_RX Capture Function
0
1
2
RXDFE
ACMP0
ACMP1
UART0_RX
From Internal or External Reference Voltage

Figure 3-4. IR demodulation diagram

3.2.7 RTC capture

RTC overflow may be captured by FTM0 channel 0 by setting SIM_SOPT[FTMIC]. When this field is correctly set, the RTC overflow is connected to FTM0 channel 0 for capture, the FTM0_CH0 pin is released to other shared functions.

3.2.8 FTM2 software synchronization

FTM2 contains three synchronization input triggers, one of which is a software trigger by writing 1 to SIM_SOPT[FTMSYNC]. Writing 0 to this field takes no effect. This field is always read 0.

3.2.9 ADC hardware trigger

ADC module may initiate a conversion via a hardware trigger. The following table shows the available ADC hardware trigger sources by setting SIM_SOPT[ADHWT].

Table 3-3. ADC hardware trigger setting

ADHWT	ADC hardware trigger
000	RTC overflow
001	FTM0 init trigger
010	FTM2 init trigger with 8-bit programmable delay
011	FTM2 match trigger with 8-bit programmable delay
100	PIT ch0 overflow
101	PIT ch1 overflow
110	ACMP0 out
111	ACMP1 out

When ADC hardware trigger selects the output of FTM2 triggers, an 8-bit delay block will be enabled. This logic delays any trigger from FTM2 with an 8-bit counter whose value is specified by SIM_SOPT[DELAY]. The reference clock to this module is the bus clock with selectable predivider specified by SIM_SOPT[BUSREF].

3.3 Core Modules

3.3.1 ARM Cortex-M0+ core configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at **arm.com**.

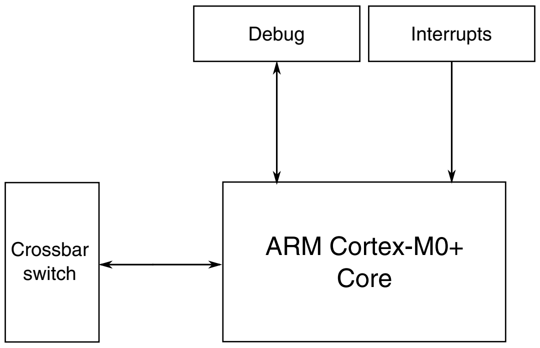


Figure 3-5. Core configuration

Table 3-4. Reference links to related information

Topic	Related module	Reference
Full description	ARM Cortex-M0+ core, r0p0	ARM Cortex-M0+ Technical Reference Manual, r0p0
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
System/instruction/data bus module	Crossbar switch	Crossbar switch
Debug	Serial Wire Debug (SWD)	Debug
Interrupts	Nested Vectored Interrupt Controller (NVIC)	NVIC
	Miscellaneous Control Module (MCM)	MCM

3.3.1.1 ARM Cortex M0+ core

The ARM Cortex M0+ parameter settings are as follows:

Table 3-5. ARM Cortex-M0+ parameter settings

Parameter	Verilog name	Value	Description
Arch Clock Gating	ACG	1 = Present	Implements architectural clock gating
DAP Slave Port Support	AHBSLV	1	Supports any AHB debug access port (like the CM4 DAP)
DAP ROM Table Base	BASEADDR	0xF000_2003	Base address for DAP ROM table
Endianess	BE	0	Little endian control for data transfers
Breakpoints	BKPT	2	Implements 2 breakpoints
Debug Support	DBG	1 = Present	—

Table continues on the next page...

Table 3-5. ARM Cortex-M0+ parameter settings (continued)

Parameter	Verilog name	Value	Description
Halt Event Support	HALTEV	1 = Present	—
I/O Port	IOP	1 = Present	Implements single-cycle ld/st accesses to special addr space
IRQ Mask Enable	IRQDIS	0x0	—
Debug Port Protocol	JTAGnSW	0 = SWD	SWD protocol, not JTAG
Core Memory Protection	MPU	0 = Absent	No MPU
Number of IRQs	NUMIRQ	32	Assume full NVIC request vector
Reset all regs	RAR	0 = Standard	Do not force all registers to be async reset
Multiplier	SMUL	0 = Fast Mul	Implements single-cycle multiplier
Multi-drop Support	SWMD	0 = Absent	Do not include serial wire support for multi-drop
System Tick Timer	SYST	1 = Present	Implements system tick timer (for CM4 compatibility)
DAP Target ID	TARGETID	0	—
User/Privileged	USER	1 = Present	Implements processor operating modes
Vector Table Offset Register	VTOR	1 = Present	Implements relocation of exception vector table
WIC Support	WIC	1 = Present	Implements WIC interface
WIC Requests	WICLINES	34	Exact number of wakeup IRQs is 34
Watchpoints	WPT	2	Implements 2 watchpoints

For details on the ARM Cortex-M0+ processor core see the ARM website: **arm.com**.

3.3.1.2 Buses, interconnects, and interfaces

The ARM Cortex-M0+ core has two bus interfaces:

- Single 32-bit AMBA-3 AHB-Lite system interface that provides connections to peripherals and all system memory, which includes flash and RAM.
- Single 32-bit I/O port bus (IOPORT) interfacing to the FGPIO with 1-cycle loads and stores.

3.3.1.3 System Tick Timer

The CLKSOURCE field in SysTick Control and Status register selects either the core clock (when CLKSOURCE = 1) or a divide-by-16 of the core clock (when CLKSOURCE = 0). Because the timing reference is a variable frequency, the TENMS field in the SysTick Calibration Value Register is always zero.

3.3.1.4 Core privilege levels

The core on this device is implemented with both privileged and unprivileged levels. The ARM documentation uses different terms than this document to distinguish between privilege levels.

If you see this term...	it also means this term...
Privileged	Supervisor
Unprivileged or user	User

3.3.1.5 Caches

This device does not have processor related cache memories, but the flash controller has an internal 32-byte cache for flash access.

3.3.2 Nested Vectored Interrupt Controller (NVIC) configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at **arm.com**.

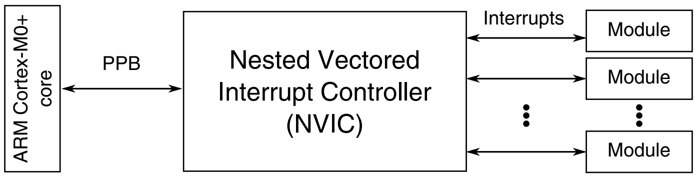


Figure 3-6. NVIC configuration

Table 3-6. Reference links to related information

Topic	Related module	Reference
Full description	Nested Vectored Interrupt Controller (NVIC)	ARM Cortex-M0+ Technical Reference Manual
System memory map	—	System memory map
Clocking	—	Clock distribution
Power management		Power management
Private Peripheral Bus (PPB)	ARM Cortex-M0+ core	ARM Cortex-M0+ core

3.3.2.1 Interrupt priority levels

This device supports four priority levels for interrupts. Therefore in the NVIC, each source in the IPR registers contains two bits. For example, IPR0 is shown below:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQ3		0	0	0	0	0	0	IRQ2		0	0	0	0	0	0	IRQ1		0	0	0	0	0	0	IRQ0		0	0	0	0	0	0
W																																

3.3.2.2 Non-maskable interrupt

The non-maskable interrupt request to the NVIC is controlled by the external $\overline{\text{NMI}}$ signal. The pin, which the $\overline{\text{NMI}}$ signal is multiplexed on, must be configured for the $\overline{\text{NMI}}$ function to generate the non-maskable interrupt request.

3.3.2.3 Interrupt channel assignments

The interrupt vector assignments are defined in the following table.

- Vector number — the value stored on the stack when an interrupt is serviced.
- IRQ number — non-core interrupt source count, which is the vector number minus 16.

The IRQ number is used within ARM's NVIC documentation.

Table 3-8. Interrupt vector assignments

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
ARM Core System Handler Vectors					
0x0000_0000	0	—	—	ARM core	Initial Stack Pointer
0x0000_0004	1	—	—	ARM core	Initial Program Counter
0x0000_0008	2	—	—	ARM core	Non-maskable Interrupt (NMI)
0x0000_000C	3	—	—	ARM core	Hard Fault
0x0000_0010	4	—	—	—	—
0x0000_0014	5	—	—	—	—
0x0000_0018	6	—	—	—	—
0x0000_001C	7	—	—	—	—

Table continues on the next page...

Table 3-8. Interrupt vector assignments (continued)

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
0x0000_0020	8	—	—	—	—
0x0000_0024	9	—	—	—	—
0x0000_0028	10	—	—	—	—
0x0000_002C	11	—	—	ARM core	Supervisor call (SVCall)
0x0000_0030	12	—	—	—	—
0x0000_0034	13	—	—	—	—
0x0000_0038	14	—	—	ARM core	Pendable request for system service (PendableSrvReq)
0x0000_003C	15	—	—	ARM core	System tick timer (SysTick)
Non-Core Vectors					
0x0000_0040	16	0	0	—	—
0x0000_0044	17	1	0	—	—
0x0000_0048	18	2	0	—	—
0x0000_004C	19	3	0	—	—
0x0000_0050	20	4	1	—	—
0x0000_0054	21	5	1	FTMRE	Command complete
0x0000_0058	22	6	1	PMC	Low-voltage warning
0x0000_005C	23	7	1	IRQ	External interrupt
0x0000_0060	24	8	2	I^2C0	Single interrupt vector for all sources
0x0000_0064	25	9	2	—	—
0x0000_0068	26	10	2	SPI0	Single interrupt vector for all sources
0x0000_006C	27	11	2	—	
0x0000_0070	28	12	3	UART0	Status and error
0x0000_0074	29	13	3	—	
0x0000_0078	30	14	3	—	
0x0000_007C	31	15	3	ADC0	ADC conversion complete interrupt
0x0000_0080	32	16	4	ACMP0	Analog comparator 0 interrupt
0x0000_0084	33	17	4	FTM0	Single interrupt vector for all sources
0x0000_0088	34	18	4	—	
0x0000_008C	35	19	4	FTM2	Single interrupt vector for all sources
0x0000_0090	36	20	5	RTC	RTC overflow
0x0000_0094	37	21	5	ACMP1	Analog comparator 1 interrupt
0x0000_0098	38	22	5	PIT_CH0	PIT CH0 overflow
0x0000_009C	39	23	5	PIT_CH1	PIT CH1 overflow
0x0000_00A0	40	24	6	KBI0	Keyboard interrupt0
0x0000_00A4	41	25	6	KBI1	Keyboard interrupt1
0x0000_00A8	42	26	6	—	
0x0000_00AC	43	27	6	ICS	Clock loss of lock

Table continues on the next page...

Table 3-8. Interrupt vector assignments (continued)

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
0x0000_00B0	44	28	7	WDOG	Watchdog timeout
0x0000_00B4	45	29	7	PWT	Single interrupt vector for all sources
0x0000_00B8	46	30	7	—	
0x0000_00BC	47	31	7	—	

1. Indicates the NVIC's interrupt source number.
2. Indicates the NVIC's IPR register number used for this IRQ. The equation to calculate this value is: IRQ div 4

3.3.2.3.1 Determining the field and register location for configuring a particular interrupt

Suppose you need to configure the SPI0 interrupt. The following table is an excerpt of the SPI0 row from Interrupt priority levels.

Table 3-9. Interrupt vector assignments

Address	Vector	IRQ[1]	NVIC IPR register number[2]	Source module	Source description
0x0000_0068	26	10	2	SPI0	Single interrupt vector for all sources

1. Indicates the NVIC's interrupt source number.
2. Indicates the NVIC's IPR register number used for this IRQ. The equation to calculate this value is: IRQ div 4.

- The NVIC registers you would use to configure the interrupt are:
  - NVICIPR2
- To determine the particular IRQ's field location within these particular registers:
  - NVICIPR2 field starting location = 8 * (IRQ mod 4) + 6 = 22

  Since the NVICIPR fields are 2-bit wide (4 priority levels), the NVICIPR2 field range is bits 22–23.

Therefore, the field locations NVICIPR2[23:22] are used to configure the SPI0 interrupts.

3.3.3 Asynchronous wakeup interrupt controller (AWIC) configuration

This section summarizes how the module has been configured in the chip. Full documentation for this module is provided by ARM and can be found at **arm.com**.

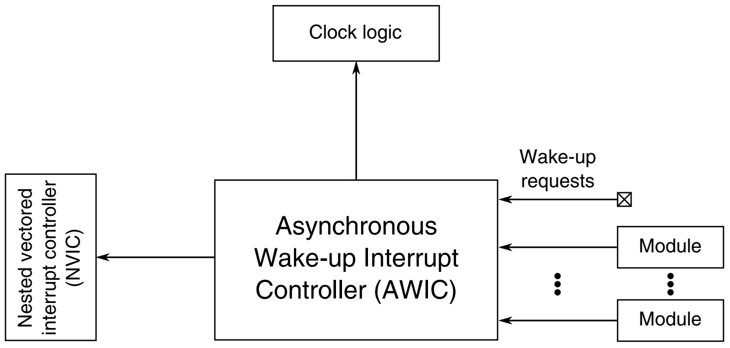

Clock logic
Wake-up requests
Nested vectored interrupt controller (NVIC)
Asynchronous Wake-up Interrupt Controller (AWIC)
Module
Module

Figure 3-7. Asynchronous wake-up interrupt controller configuration

Table 3-10. Reference links to related information

Topic	Related module	Reference
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Interrupt control	Nested vectored interrupt controller (NVIC)	NVIC
Wake-up requests		AWIC wake-up sources

3.3.3.1 Wakeup sources

The device uses the following internal and external inputs to the AWIC module.

Table 3-11. AWIC stop wakeup sources

Wake-up source	Description
Available system resets	$\overline{\text{RESET}}$ pin when LPO is its clock source
Low-voltage warning	Power management controller
IRQ	IRQ pin
Pin interrupts	KBI - Any enabled pin interrupt is capable of waking the system.
ADC	The ADC is functional in Stop mode when using internal clock source.
ACMP	Interrupt in normal
I^2C	Address match wake-up
SPI	SPI slave mode interrupt
UART	UART active edge detect at UART_RX pin
RTC	Alarm interrupt
Non-maskable interrupt	$\overline{\text{NMI}}$ pin

3.4 System Modules

3.4.1 SIM configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module’s dedicated chapter.

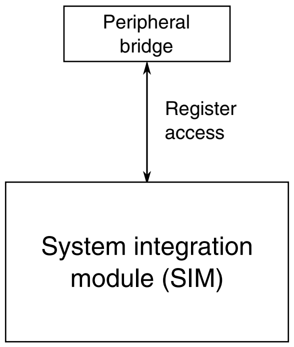


Figure 3-8. SIM configuration

Table 3-12. Reference links to related information

Topic	Related module	Reference
Full description	SIM	SIM
System memory map	—	System memory map
Clocking	—	Clock distribution
Power management	—	Power management

3.4.2 PMC configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module’s dedicated chapter.

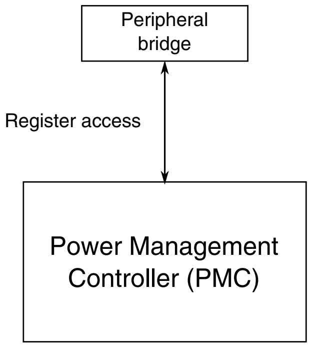


Figure 3-9. PMC configuration

Table 3-13. Reference links to related information

Topic	Related module	Reference
Full description	PMC	PMC
System memory map	—	System memory map
Clocking	—	Clock distribution

3.4.3 MCM configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

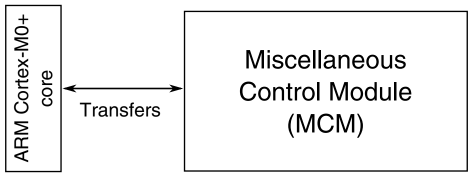


Figure 3-10. MCM configuration

Table 3-14. Reference links to related information

Topic	Related module	Reference
Full description	Miscellaneous Control module (MCM)	MCM
System memory map	—	System memory map
Clocking	—	Clock distribution
Power management	—	Power management
Private Peripheral Bus (PPB)	ARM Cortex-M0+ core	ARM Cortex-M0+ core
Transfer	Flash memory controller	Flash memory controller

3.4.4 Crossbar-light switch configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

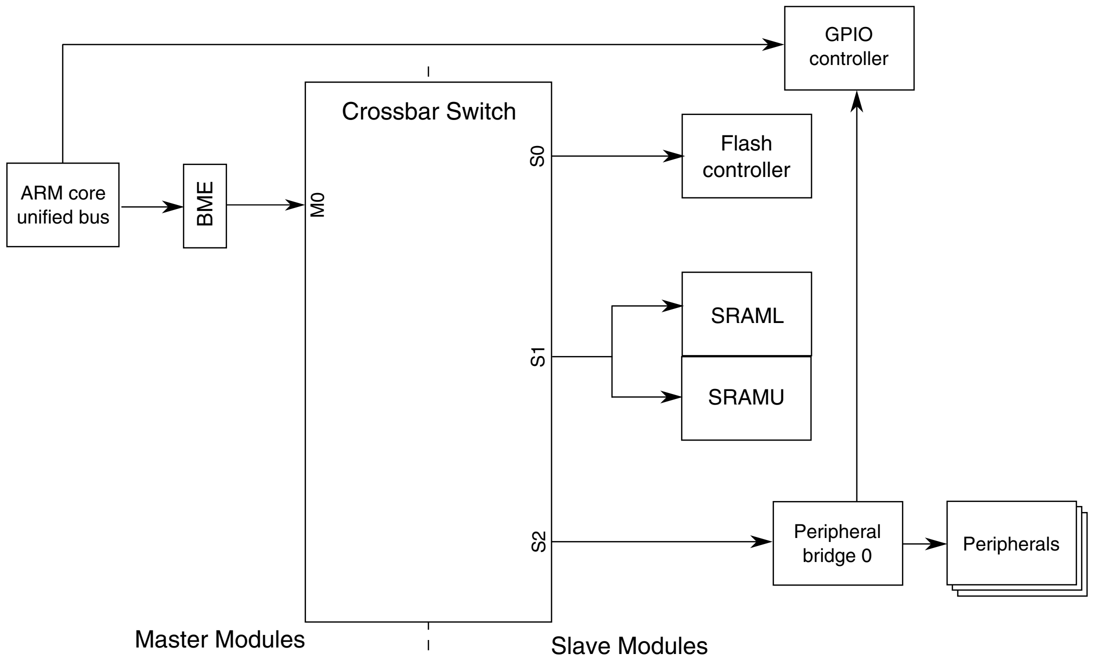


Figure 3-11. Crossbar-Light switch integration

Table 3-15. Reference links to related information

Topic	Related module	Reference
System memory map	—	System memory map
Clocking	—	Clock Distribution
Crossbar switch master	ARM Cortex-M0+ core	ARM Cortex-M0+ core
Crossbar switch slave	Flash memory controller	Flash memory controller
Crossbar switch slave	SRAM controller	SRAM configuration
Crossbar switch slave	Peripheral bridge	Peripheral bridge
2-ported peripheral	GPIO controller	GPIO controller

3.4.4.1 Crossbar-Light switch master assignments

The masters connected to the crossbar switch are assigned as follows:

Master module	Master port number
ARM core unified bus	0

3.4.4.2 Crossbar switch slave assignments

This device contains 3 slaves connected to the crossbar switch.

The slave assignment is as follows:

Slave module	Slave port number
Flash memory controller	0
SRAM controller	1
Peripheral bridge	2

3.4.5 Peripheral bridge configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

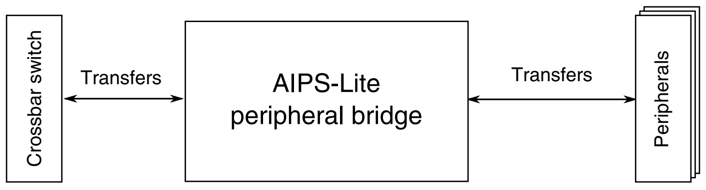


Figure 3-12. Peripheral bridge configuration

Table 3-16. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge (AIPS-Lite)	Peripheral bridge (AIPS-Lite)
System memory map	—	System memory map
Clocking	—	Clock distribution

3.4.5.1 Number of peripheral bridges

This device contains one peripheral bridge.

3.4.5.2 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this device. See AIPS-Lite Memory Map for the memory slot assignment for each module.

3.5 System Security

3.5.1 CRC configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

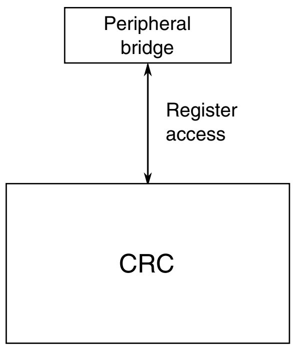


Figure 3-13. CRC configuration

Table 3-17. Reference links to related information

Topic	Related module	Reference
Full description	CRC	CRC
System memory map	—	System memory map
Power management	—	Power management

3.5.2 Watchdog configuration

This section summarizes how the module has been configured in the chip.

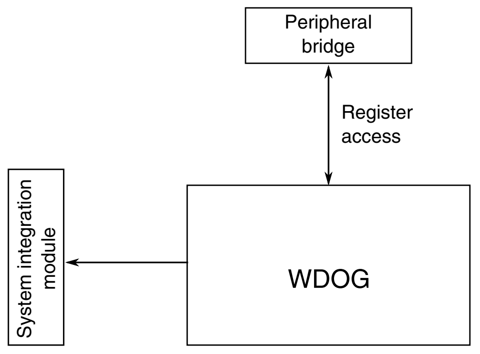


Figure 3-14. Watchdog configuration

Table 3-18. Reference links to related information

Topic	Related module	Reference
Full description	Watchdog (WDOG)	Watchdog (WDOG)
Clocking	—	Clock distribution
Power management	—	Power management
Programming model	System Integration Module (SIM)	SIM

3.5.2.1 WDOG clocks

The watchdog has four selectable clock sources:

- 1 kHz internal low power oscillator (LPOCLK)
- Internal 32 kHz reference clock (ICSIRCLK)
- External clock (OSCERCLK)
- Bus clock

3.5.2.2 WDOG operation

The WDOG module provides a fail-safe mechanism to ensure the system can be reset to a known state of operation in case of system failure, such as the CPU clock stopping or there being a run away condition in the software code. The watchdog counter runs continuously off a selectable clock source and expects to be serviced (refreshed) periodically. If it is not, it resets the system.

After any reset, the WDOG watchdog is enabled. If the WDOG watchdog is not used in an application, it can be disabled by clearing WDOG_CS1[EN].

The refresh write sequence is a write of 0xA602 followed by a write of 0xB480 to the WDOG_CNTH:L registers. The write of the 0xB480 must occur within 16 bus clocks after the write of 0xA602; otherwise, the watchdog resets the MCU.

The watchdog counter has four clock source options selected by programming WDOG_CS2[CLK]. The clock source options are the bus clock, internal 1 kHz clock, external clock, or an internal 32 kHz clock source.

The refresh timeout time is defined by WDOG_TOVALH:L. In addition, if window mode is used, software must not start the refresh sequence until after the time value set in the WDOG_WINH:L registers.

An optional fixed prescaler for all clock sources allows for longer timeout periods. When WDOG_CS2[PRES] is set, the clock source is prescaled by 256 before clocking the watchdog counter.

The watchdog counter registers CNTH:L provide access to the value of the free-running watchdog counter. The software can read the counter registers at any time but cannot write directly to the watchdog counter. The refresh sequence resets the watchdog counter to 0x0000. Write to the WDOG_CNTH:L registers of 0xC520 followed by 0xD928 within 16 bus clocks start the unlock sequence. On completing the unlock sequence, the user must reconfigure the watchdog within 128 bus clocks; otherwise, the watchdog forces a reset to the MCU.

By default, the watchdog is not functional in Debug mode, Wait mode, or Stop mode. Setting WDOG_CS1[DBG], WDOG_CS1[WAIT] or WDOG_CS1[STOP] can activate the watchdog in Debug, Wait or Stop modes.

3.6 Clock Modules

3.6.1 ICS configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

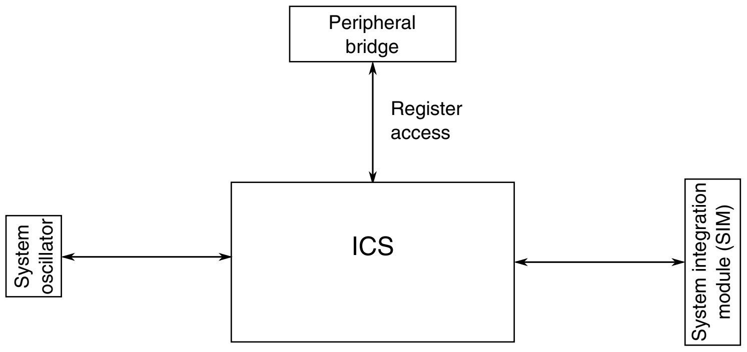


Figure 3-15. ICS configuration

Table 3-19. Reference links to related information

Topic	Related module	Reference
Full description	ICS	ICS
System memory map	—	System memory map
Clocking	—	Clock distribution
Power management	—	Power management

3.6.1.1 Clock gating

This family of devices includes clock gating control for each peripheral, that is, the clock to each peripheral can explicitly be gated on or off, using clock-gate control bits in the SIM_SCGC register.

3.6.2 OSC configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

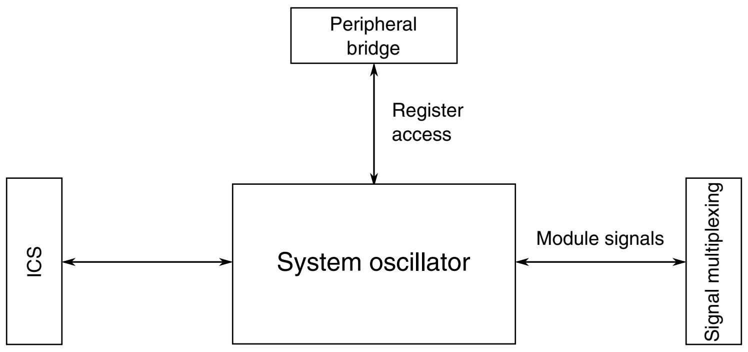


Figure 3-16. OSC configuration

Table 3-20. Reference links to related information

Topic	Related module	Reference
Full description	OSC	OSC
System memory map	—	System memory map
Clocking	—	Clock distribution
Power management	—	Power management
Full description	ICS	ICS

3.7 Memories and Memory Interfaces

3.7.1 Flash memory configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

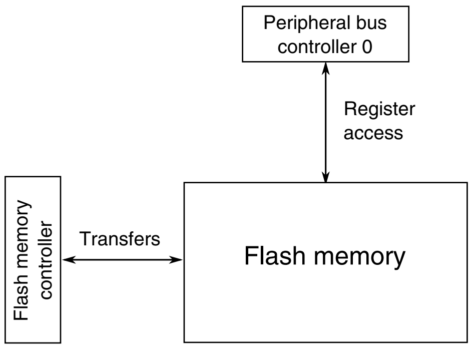


Figure 3-17. Flash memory configuration

Table 3-21. Reference links to related information

Topic	Related module	Reference
Full description	Flash memory	Flash memory
System memory map	—	System memory map
Clocking	—	Clock Distribution
Transfers	Flash memory controller	Flash memory controller
Register access	Peripheral bridge	Peripheral bridge

3.7.1.1 Flash memory sizes

The devices covered in this document contain one flash block consisting of up to 16 sectors of 512 bytes.

The amounts of flash memory for the devices covered in this document are:

Table 3-22. KE04 flash memory size

Device	EEPROM (B)	Flash (KB)	Block 0 (flash) address range
MKE04Z8VTG4(R)	0	8	0x0000_0000 – 0x0000_1FFF
MKE04Z8VWJ4(R)	0	8	0x0000_0000 – 0x0000_1FFF
MKE04Z8VFK4(R)	0	8	0x0000_0000 – 0x0000_1FFF

3.7.1.2 Flash memory map

The flash memory and the flash registers are located at different base addresses as shown in the figure found here.

The base address for each is specified in System memory map.

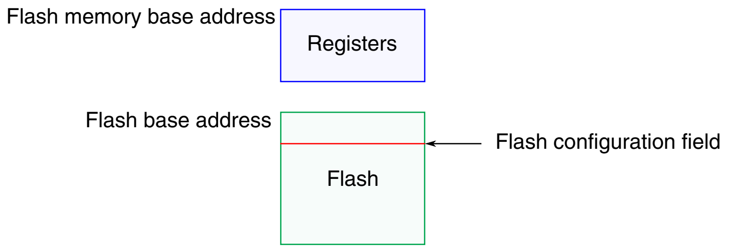


Figure 3-18. Flash memory map

The on-chip flash memory is implemented in a portion of the allocated Flash range to form a contiguous block in the memory map beginning at address 0x0000_0000. See Flash memory sizes for details of supported ranges.

Access to the flash memory ranges outside the amount of flash on the device causes the bus cycle to be terminated with an error followed by the appropriate response in the requesting bus master.

3.7.1.3 Flash security

For information on how flash security is implemented on this device, see Chip Security.

3.7.1.4 Erase all flash contents

In addition to software, the entire flash memory may be erased external to the flash memory via the SW-DP debug port by setting MDM-AP CONTROL[0] (bit 0 of the MDM-AP Control register). MDM-AP STATUS[0] (bit 0 of the MDM-AP Status register) is set to indicate the mass erase command has been accepted. MDM-AP CONTROL[0] is cleared when the mass erase completes.

3.7.2 Flash memory controller configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

See Platform Control Register (MCM_PLACR) register description for details on the reset configuration of the FMC.

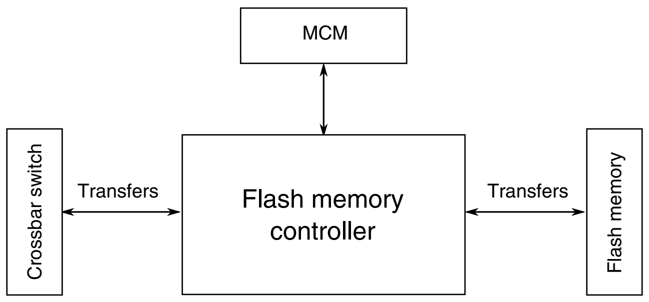


Figure 3-19. Flash memory controller configuration

Table 3-23. Reference links to related information

Topic	Related module	Reference
Full description	Flash memory controller	Flash memory controller
System memory map		System memory map
Clocking		Clock Distribution
Transfers	Flash memory	Flash memory
Transfers	Crossbar switch	Crossbar Switch
Register access	MCM	MCM

3.7.3 SRAM configuration

This section summarizes how the module has been configured in the chip.

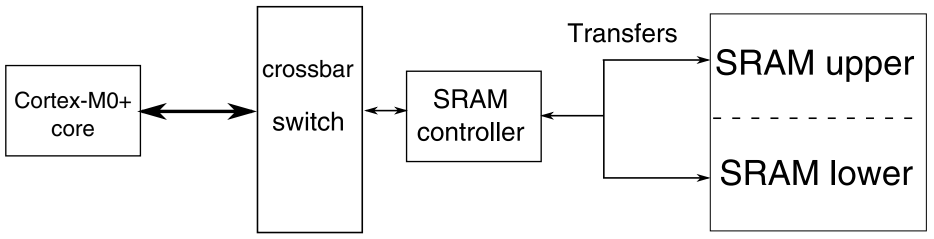


Figure 3-20. SRAM configuration

Table 3-24. Reference links to related information

Topic	Related module	Reference
Full description	SRAM	SRAM
System memory map	—	System memory map
Clocking	—	Clock Distribution
ARM Cortex-M0+ core	—	ARM Cortex-M0+ core

3.7.3.1 SRAM sizes

The SRAM supports single cycle access (zero wait states) at all core speeds.

The amounts of SRAM for the devices covered in this document are:

Table 3-25. SRAM size

Freescale part number	SRAM
MKE04Z8VTG4(R)	1 KB
MKE04Z8VWJ4(R)	1 KB
MKE04Z8VFK4(R)	1 KB

3.7.3.2 SRAM ranges

The on-chip SRAM is split into two ranges; 1/4 is allocated to SRAM_L and 3/4 is allocated to SRAM_U.

The on-chip RAM is implemented such that the SRAM_L and SRAM_U ranges form a contiguous block in the memory map. As such:

- SRAM_L is anchored to 0x1FFF_FFFF and occupies the space before this ending address.
- SRAM_U is anchored to 0x2000_0000 and occupies the space after this beginning address.

Valid address ranges for SRAM_L and SRAM_U are then defined as:

- SRAM_L = [0x2000_0000–(SRAM_size/4)] to 0x1FFF_FFFF
- SRAM_U = 0x2000_0000 to [0x2000_0000+(SRAM_size*(3/4))-1]

This is illustrated in the following figure.

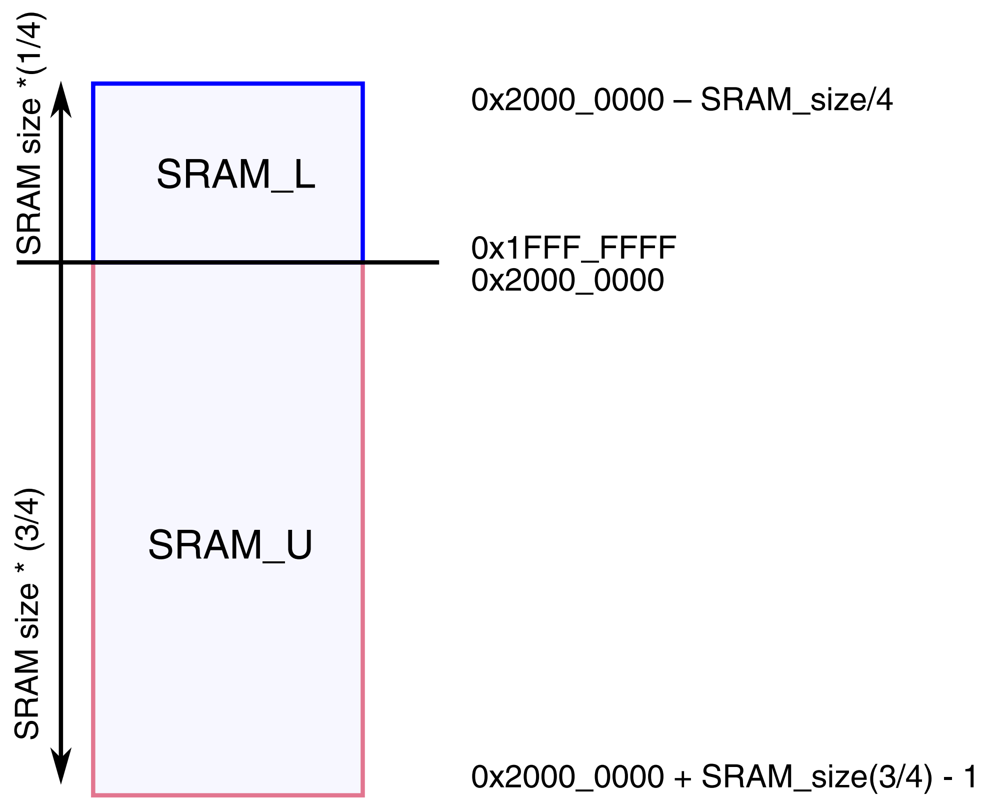


Figure 3-21. SRAM blocks memory map

For example, for a device containing 1 KB of SRAM, the ranges are:

- SRAM_L: 0x1FFF_FF00 – 0x1FFF_FFFF
- SRAM_U: 0x2000_0000 – 0x2000_02FF

3.7.3.3 SRAM bit operation

The on-chip SRAM is split to two range: SRAM_L and SRAM_U. The SRAM_U range supports bit operation on this device through two ways:

- Aliased bit-band region
- Bit Manipulation Engine (BME)

A 32-bit write in the aliased region has the same effect as a read-modify-write operation on the targeted bit in the SRAM_U region. See Aliased bit-band region for details.

The aliased bit-band region only supports simple set or clear operation. More complicated bit operations (AND, OR, XOR, etc) could be further supported through the BME engine. See Bit Manipulation Engine for details.

3.8 Analog

3.8.1 12-bit analog-to-digital converter (ADC) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

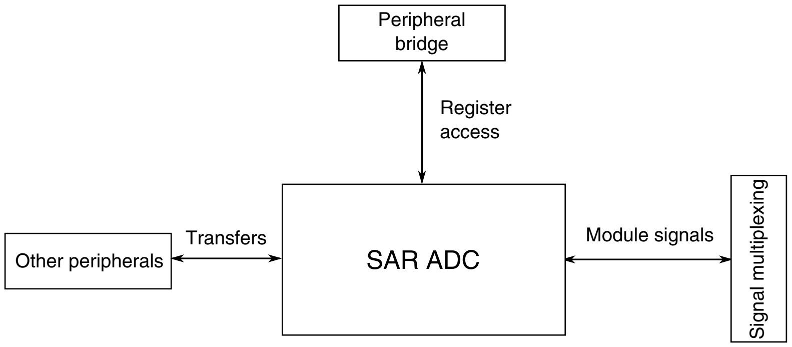


Figure 3-22. 12-bit SAR ADC configuration

Table 3-26. Reference links to related information

Topic	Related module	Reference
Full description	12-bit SAR ADC	12-bit SAR ADC
System memory map	—	System memory map
Clocking	—	Clock distribution
Power management	—	Power management

3.8.1.1 ADC instantiation information

This device contains one 12-bit successive approximation ADC with up to 12 channels.

Table 3-27. ADC channels

Freescale part number	ADC channels
MKE04Z8VTG4(R)	6
MKE04Z8VWJ4(R)	10
MKE04Z8VFK4(R)	12

The ADC supports both software and hardware triggers. The ADC hardware trigger, ADHWT, is selectable from ACMP0, ACMP1, FTM0 init trigger, FTM2 init trigger, FTM2 match trigger, RTC overflow, or PITCH0/1 overflow. The hardware trigger can be configured to cause a hardware trigger in MCU Run, Wait, and Stop modes.

The hardware trigger sources details are listed in the Module-to-Module section.

3.8.1.2 ADC0 connections/channel assignment

The ADC channel assignments for the device are shown in following table. Reserved channels convert to an unknown value.

Table 3-28. ADC channel assigement

ADCH	Channel	Input
00000	AD0	PTA0/ADP0
00001	AD1	PTA1/ADP1
00010	AD2	PTA6/ADP2
00011	AD3	PTA7/ADP3
00100	AD4	PTB0/ADP4
00101	AD5	PTB1/ADP5
00110	AD6	PTB2/ADP6
00111	AD7	PTB3/ADP7
01000	AD8	PTC0/ADP8
01001	AD9	PTC1/ADP9
01010	AD10	PTC2/ADP10
01011	AD11	PTC3/ADP11
01100	AD12	Vss
01101	AD13	Vss
01110	AD14	Vss
01111	AD15	Vss
10000	AD16	Vss
10001	AD17	Vss
10010	AD18	Vss
10011	AD19	Vss
10100	AD20	Reserved
10101	AD21	Reserved
10110	AD22	Temperature Sensor
10111	AD23	Bandgap
11000	AD24	Reserved
11001	AD25	Reserved
11010	AD26	Reserved

Table continues on the next page...

Table 3-28. ADC channel assigement (continued)

ADCH	Channel	Input
11011	AD27	Reserved
11100	AD28	Reserved
11101	AD29	VREFH
11110	AD30	VREFL
11111	Module disabled	None

3.8.1.3 ADC analog supply and reference connections

This device only includes VDD and VSS pin at chip package.

The VDDA and VREFH of ADC are internally connected to VDD pin. The VSSA and VREFL of ADC are internally connected to VSS pin.

3.8.1.4 Temperature sensor and bandgap

The ADC module integrates an on-chip temperature sensor. Following actions must be performed to use this temperature sensor.

- Configure ADC for long sample with a maximum of 1 MHz clock
- Convert the bandgap voltage reference channel (AD23)
  - By converting the digital value of the bandgap voltage reference channel using the value of V_{BG}, the user can determine V_{DD}.
- Convert the temperature sensor channel (AD22)
  - By using the calculated value of V_{DD}, convert the digital value of AD22 into a voltage, V_{TEMP}

3.8.1.5 Alternate clock

The ADC module is capable of performing conversions using the MCU bus clock, the bus clock divided by 2, the local asynchronous clock (ADACK) within the module, or the alternate clock, ALTCLK. The alternate clock for the devices is the external oscillator output (OSC_OUT).

The selected clock source must run at a frequency such that the ADC conversion clock (ADCK) runs at a frequency within its specified range (f_{ADCK}) after being divided down from the ALTCLK input as determined by ADC_SC3[ADIV].

ALTCLK is active while the MCU is in Wait mode provided the conditions described above are met. This allows ALTCLK to be used as the conversion clock source for the ADC while the MCU is in Wait mode.

ALTCLK cannot be used as the ADC conversion clock source while the MCU is in Stop mode.

3.8.2 ACMP configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

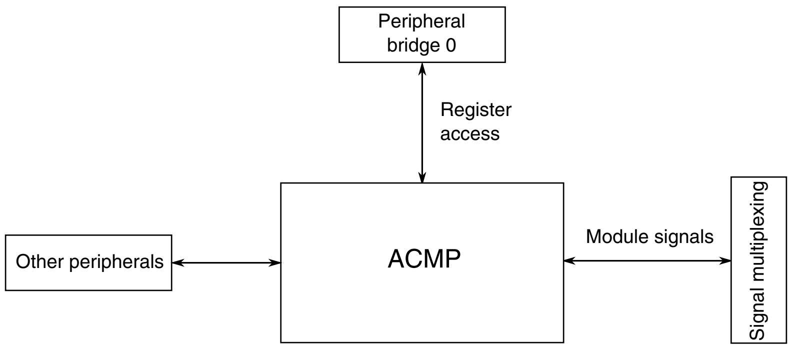


Figure 3-23. ACMP configuration

Table 3-29. Reference links to related information

Topic	Related module	Reference
Full description	Analog comparator (ACMP)	Comparator
System memory map	—	System memory map
Clocking	—	Clock distribution
Power management	—	Power management

3.8.2.1 ACMP overview

The device contains two analog comparator modules (ACMP) which provide a circuit for comparing two analog input voltages or for comparing one analog input voltage to an internal reference voltage. The comparator circuit is used to operate across the full range of the supply voltage (rail-to-rail operation).

The ACMP features four different inputs muxed with both positive and negative inputs to the ACMP. One is fixed connected to built-in DAC output, the others are externally mapped on pinouts.

The ACMP modules support internal bandgap reference voltage. When using the bandgap reference, the user must enable the PMC bandgap buffer first.

The ACMP modules can continue to operate in Wait and Stop mode if enabled, and can wake the MCU when a compare event occurs.

3.8.2.2 ACMP interconnections

The ACMP0 or ACMP1 output can be configured to connect with FTM0 input capture channel 0 by setting SIM_SOPT[FTMIC] with correct value. Writing any value other than 00b to SIM_SOPT[FTMIC], the FTM0_CH0 pin is not available externally regardless of the configuration of the FTM0 module for channel 0.

ACMP0 and ACMP1 output are also internal connected to FTM2 trigger input and Fault input. By configuring SIM_SOPT[ACTRG] user can connect either ACMP0_OUT or ACMP1_OUT to FTM2 trigger input 0. ACMP0_OUT is connected to FTM2 fault input 0 and ACMP1_OUT is connected to FTM2 fault input 3.

ACMP0 or ACMP1 output can be directly ejected to UART0_RX by setting SIM_SOPT[RXDCE]. In this mode, UART0_RX pinout does not work. Any external signal tagged to ACMP0 or ACMP1 inputs can be regarded as input pins.

The following table shows the input connections to the ACMP0 and ACMP1:

Table 3-30. ACMP0 input connections

ACMP0 channel	Connection
0	PTB2/ACMP0_IN0
1	PTA1/ACMP0_IN1
2	PTA0/ACMP0_IN2
3	DAC output

Table 3-31. ACMP1 input connections

ACMP1 channel	Connection
0	PTA6/ACMP1_IN0
1	PTA7/ACMP1_IN1
2	PTB4/ACMP1_IN2
3	DAC output

3.8.2.3 ACMP in Stop mode

ACMP continues to operate in Stop mode if enabled. If ACMPx_SC[ACOPE] is enabled, comparator output will operate as in the normal operating mode and will control ACMPx_OUT pin. The MCU is brought out of Stop mode when a compare event occurs and ACMPx_CS[ACIE] is enabled; ACMPx_CS[ACF] flag sets accordingly.

3.9 Timers

3.9.1 FlexTimer configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

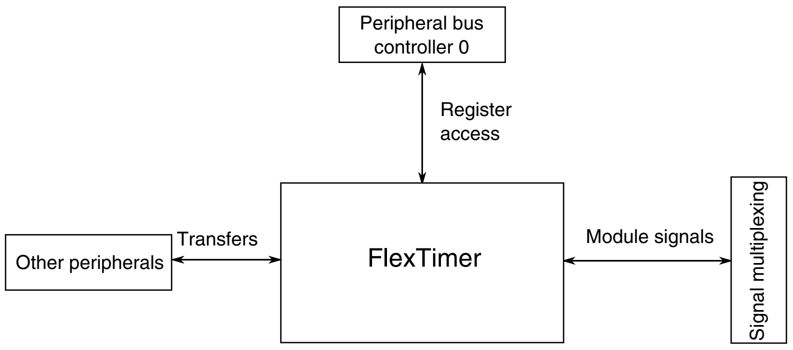


Figure 3-24. FlexTimer configuration

Table 3-32. Reference links to related information

Topic	Related module	Reference
Full description	FlexTimer	FlexTimer
System memory map	—	System memory map
Clocking	—	Clock distribution
Power management	—	Power management
Signal multiplexing	Port control	Signal multiplexing

3.9.1.1 FTM overview

The FTM timer contains up to six channels which support input capture, output compare and the generation of PWM signals to control electric motor and power management applications. FTM time reference is a 16-bit counter which can be used as an unsigned or signed counter.

This device contains up to two FTM modules of one 6-channel FTM with full functions and one 2-channel FTM with basic TPM functions. Each FTM module can use independent external clock input. The table below summarizes the configuration of FTM modules.

Table 3-33. FTM modules features

Feature	FTM0	FTM2
Number of channels	2	6
Initial counting value	no	yes
Periodic TOF	no	yes
Input capture mode	yes	yes
Channel input filter	no	channels 0, 1, 2 and 3
Output compare mode	yes	yes
Edge-Aligned PWM (EPWM)	yes	yes
Center-Aligned PWM (CPWM)	yes	yes
Combine mode	no	yes
Complementary mode	no	yes
PWM synchronization	no	yes
Inverting	no	yes
Software output control (SWOC)	no	yes
Deadtime insertion	no	yes
Output mask	no	yes
Fault control	no	yes
Number of fault inputs	0	4
Fault input filter	no	fault inputs 0, 1 ,2 and 3
Polarity control	no	yes
Initialization	no	yes
Channel match trigger	no	yes
Initialization trigger	yes	yes
Capture test mode	no	yes
DMA	no	no
Dual edge capture mode	no	yes
Quadrature decoder mode	no	no
Quadrature decoder input filter	no	no
Debug modes	no	yes
Intermediary load	no	yes
Global time base enable[1]	no	yes
Registers available	FTM_SC, FTM_CNT, FTM_MOD, FTM_C0SC, FTM_C0V, FTM_C1SC, and FTM_C1V, FTM_EXTTRIG	FTM_SC, FTM_CNT, FTM_MOD, FTM_C0SC, FTM_C0V, FTM_C1SC, and FTM_C1V, FTM_C2SC, FTM_C2V, FTM_C3SC, FTM_C3V, FTM_C4SC, FTM_C4V, FTM_C5SC, FTM_C5V, FTM_CNTIN, FTM_STATUS, FTM_MODE, FTM_SYNC, FTM_OUTINIT, FTM_OUTMASK,

Table 3-33. FTM modules features

Feature	FTM0	FTM2
		FTM_COMBINE, FTM_DEADTIME, FTM_EXTTRIG, FTM_POL, FTM_FMS, FTM_FILTER, FTM_FLTCTRL, FTM_CONF, FTM_FLTPOL, FTM_SYNCONF, FTM_INVCTRL, FTM_SWOCTRL, and FTM_PWMLOAD

1. The global time base (GTB) feature allows the synchronization of multiple FTM modules on a chip. It requires the GTB function supported by all the related FTM modules. On this device, only one FTM module (FTM2) supports the GTB function, so the GTB function is actually not usable.

3.9.1.2 FTM clock options

The selectable FTM source clock can be the timer clock (up to 48 MHz), the fixed frequency clock, or an external clock. The selected control source is controlled by FTMx_SC[CLKS].

- When FTMx_SC[CLKS] = 00, no clock is selected (this in effect, disables the FTM counter).
- When FTMx_SC[CLKS] = 01, the timer clock is selected.
- When FTMx_SC[CLKS] = 10, the fixed frequency clock(ICSFFCLK) is selected.
- When FTMx_SC[CLKS] = 11, the external clock is selected.

3.9.1.3 FTM interconnections

FTM0 has following interconnections:

- UART0_TX signal can be modulated by FTM0 channel 0 PWM output.
- UART0_RX signal can be tagged by FTM0 channel 1 input capture function by writing 1 to SIM_SOPT[RXDCE].
- ACMP0, ACMP1 and RTC output signals can be captured by FTM0 channel 0.

FTM2 supports three PWM synchronization sources:

- Trigger0 is connected to the output of ACMP0 or ACMP1 by writing 0 or 1 to SIM_SOPT[ACTRG].
- Trigger1 is connected to FTM0 channel 0 output.
- Trigger2 is a software trigger by writing 1 to SIM_SOPT[FTMSYNC].

FTM2 supports four FTM fault sources:

- Fault 0 is connected to ACMP0 output.
- Fault1 is connected to PTA6.

- Fault 2 is connected to PTA7.
- Fault 3 is connected to ACMP1 output.

3.9.1.4 FTM interrupts

The FlexTimer has multiple sources of interrupt. However, either source can generate a single interrupt request to the interrupt controller. When an FTM interrupt occurs, read the FTM status registers to determine the exact interrupt source.

3.9.2 PIT configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

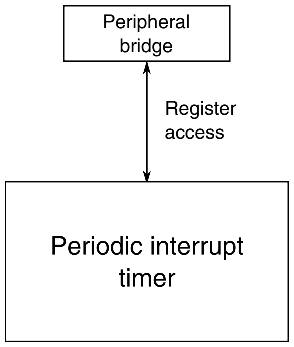


Figure 3-25. PIT configuration

Table 3-34. Reference links to related information

Topic	Related module	Reference
Full description	PIT	PIT
System memory map	—	System memory map
Clocking	—	Clock Distribution
Power management	—	Power management

3.9.2.1 PIT overview

The PIT module is an array of timers that can be used to raise interrupts and triggers.

This device contains one PIT module with two channels and supporting chained timer mode.

3.9.2.2 PIT interconnections

The PIT channel 0 and channel 1 trigger output can be used as ADC hardware trigger by setting SIM_SOPT[ADHWT].

3.9.3 RTC configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

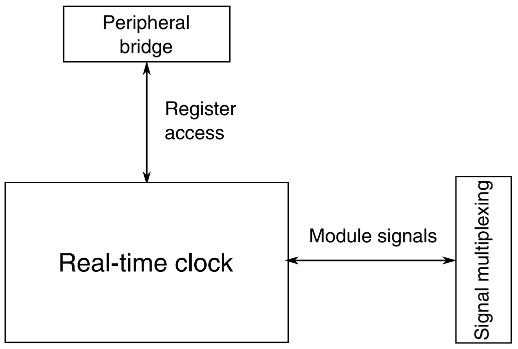


Figure 3-26. RTC configuration

Table 3-35. Reference links to related information

Topic	Related module	Reference
Full description	RTC	RTC
System memory map	—	System memory map
Clocking	—	Clock Distribution
Power management	—	Power management

3.9.3.1 RTC overview

The real-time counter (RTC) used on this device consists of one 16-bit counter, one 16-bit comparator, several binary-based and decimal-based prescaler dividers, four clock sources, and one programmable periodic interrupt. This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wake-up from low-power modes without external components.

3.9.3.2 RTC interconnections

Four software selectable clock sources are available for input to prescaler with selectable binary-based and decimal-based divider values

- 1 kHz internal low-power oscillator (LPOCLK)
- External clock (OSCERCLK)
- 32 kHz internal reference clock (ICSIRCLK)
- Bus clock

RTC overflow trigger can be used as hardware trigger for ADC and may also be captured by FTM0 channel0 through configuring SIM_SOPT register.

3.9.4 PWT configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

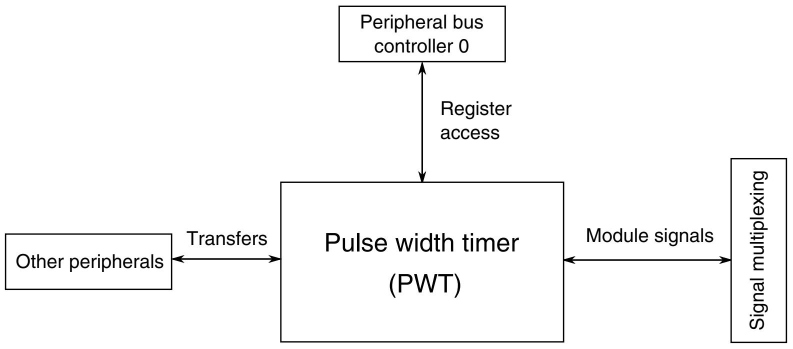


Figure 3-27. PWT configuration

Table 3-36. Reference links to related information

Topic	Related module	Reference
Full description	PWT	PWT
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

3.9.4.1 PWT overview

The Pulse Width Timer (PWT) module on this device consists of one 16-bit counter, which can be used to capture or measure the pulse width mapping on its input channels.

The counter of PWT has two selectable clocks sources, which are sharing with FTM modules, and support up to 48 MHz with internal timer clock. PWT module supports programmable positive or negative pulse edges, and programmable interrupt generation upon pulse width values or counter overflow.

3.9.4.2 PWT interconnections

Two software selectable clock sources are available for input to pre-scaler divider of PWT module:

- Timer clock : up to 48 MHz, also the option of clock source for FTM modules
- TCLK: external clock from the pads

PWT module has four input channels, which is connected as following:

Table 3-37. PWT input connections

PWT input channel	Connection
0	PTC4
1	PTB0
2	ACMP0 output
3	ACMP1 output

3.10 Communication interfaces

3.10.1 SPI configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

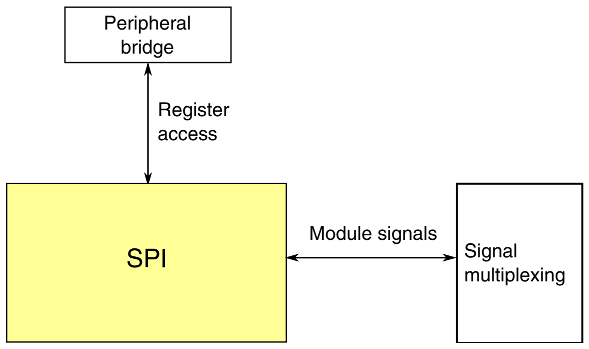


Figure 3-28. SPI configuration

Table 3-38. Reference links to related information

Topic	Related module	Reference
Full description	SPI	SPI
System memory map	—	System memory map
Clocking	—	Clock Distribution

3.10.1.1 SPI overview

This device contains one SPI module that supports 8-bit data length.

The serial peripheral interface (SPI) module provides for full-duplex, synchronous, serial communication between the MCU and peripheral devices. These peripheral devices can include other microcontrollers, analog-to-digital converters, shift registers, sensors, memories, etc.

The SPI runs at a baud rate up to the bus clock divided by two in master mode and up to the bus clock divided by 4 in slave mode. Software can poll the status flags, or SPI operation can be interrupt-driven.

3.10.2 I2C configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module’s dedicated chapter.

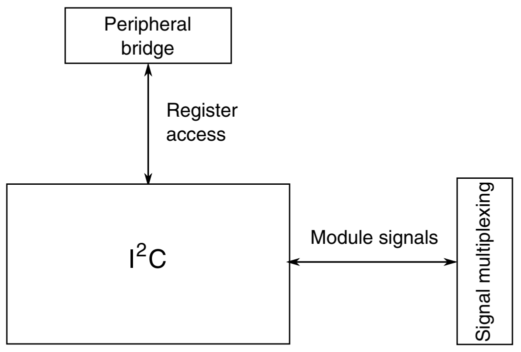


Figure 3-29. I2C configuration

Table 3-39. Reference links to related information

Topic	Related module	Reference
Full description	I^2C	I^2C
System memory map	—	System memory map
Clocking	—	Clock Distribution
Power management	—	Power management

3.10.2.1 I2C overview

This device contains an inter-integrated circuit (I2C) module providing a method of communication between a number of devices. The interface is designed to operate up to 100 kbit/s with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

3.10.3 UART configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see the module's dedicated chapter.

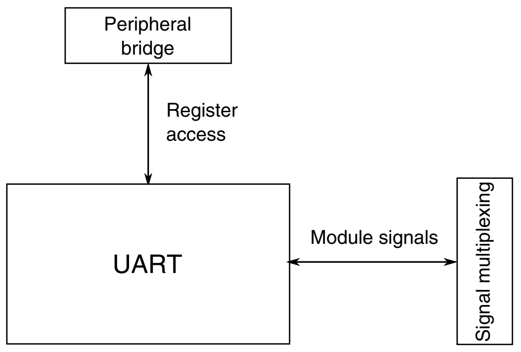


Figure 3-30. UART configuration

Table 3-40. Reference links to related information

Topic	Related module	Reference
Full description	UART	UART
System memory map	—	System memory map
Clocking	—	Clock Distribution
Power management	—	Power management

3.10.3.1 UART overview

This device includes one universal asynchronous receiver/transmitter (UART) module. Typically, these systems are used to connect to the RS232 serial input/output port of a personal computer or workstation. They can also be used to communicate with other embedded controllers.

A flexible, 13-bit, modulo-based baud rate generator supports a broad range of standard baud rates beyond 115.2 kbaud. Transmit and receive within the same UART use a common baud rate, and each UART module has a separate baud rate generator.

This UART system offers many advanced features not commonly found on other asynchronous serial I/O peripherals on other embedded controllers. The receiver employs an advanced data sampling technique that ensures reliable communication and noise detection. Hardware parity, receiver wakeup, and double buffering on transmit and receive are also included.

3.10.3.2 UART interconnection

UART0 can implement infrared functions through following tricks:

UART0_TX Modulation:

- UART0_TX output can be modulated by FTM0 channel 0 PWM output.

UART0_RX Tag:

- UART0_RX input can be tagged to FTM0 channel 1 or filtered by ACMP0 or ACMP1. module

3.11 Human-machine interfaces (HMI)

3.11.1 GPIO configuration

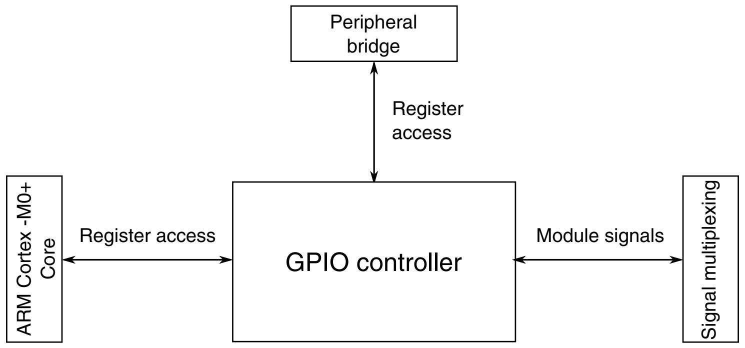


Figure 3-31. GPIO configuration

Table 3-41. Reference links to related information

Topic	Related module	Reference
Full description	GPIO	GPIO
System memory map	—	System memory map
Clocking	—	Clock Distribution
Power management	—	Power management
Crossbar switch	Crossbar switch	Crossbar switch

3.11.1.1 GPIO overview

The GPIO is multi-ported and can be accessed directly by the core with zero wait states at base address 0xF800_0000 (FGPIO). It can also be accessed by the core through the crossbar/AIPS interface at 0x400F_F000 and at an aliased slot (15) at address 0x4000_F000. All BME operations to the GPIO space can be accomplished referencing the aliased slot (15) at address 0x4000_F000. Only some of the BME operations can be accomplished referencing GPIO at address 0x400F_F000.

3.11.2 KBI configuration

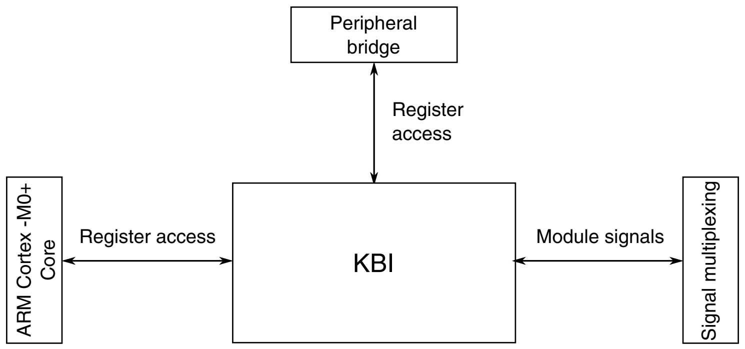


Figure 3-32. KBI configuration

Table 3-42. Reference links to related information

Topic	Related module	Reference
Full description	KBI	KBI
System memory map	—	System memory map
Clocking	—	Clock Distribution
Power management	—	Power management
Crossbar switch	Crossbar switch	Crossbar switch

3.11.2.1 KBI overview

This device has two keyboard interrupt modules (KBI) with up to 16 keyboard interrupt inputs grouped in two KBI modules available depending on package.

3.11.2.2 KBI assignments

The KBI port assignments is shown by the following table.

Table 3-43. KBI port assignment

KBI	Port pin	KBI	Port pin
KBI0_P0	PTA0	KBI1_P0	PTC4
KBI0_P1	PTA1	KBI1_P1	PTC5
KBI0_P2	PTA2	KBI1_P2	PTC0
KBI0_P3	PTA3	KBI1_P3	PTC1
KBI0_P4	PTB0	KBI1_P4	PTC2

Table continues on the next page...

Table 3-43. KBI port assignment (continued)

KBI	Port pin	KBI	Port pin
KBI0_P5	PTB1	KBI1_P5	PTC3
KBI0_P6	PTB2	KBI1_P6	PTB4
KBI0_P7	PTB3	KBI1_P7	PTB5

3.11.3 IRQ configuration

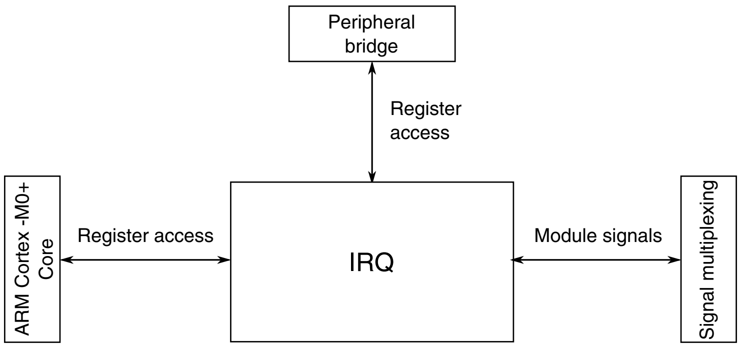


Figure 3-33. IRQ configuration

Table 3-44. Reference links to related information

Topic	Related module	Reference
Full description	IRQ	IRQ
System memory map	—	System memory map
Clocking	—	Clock Distribution
Power management	—	Power management
Crossbar switch	Crossbar switch	Crossbar switch

3.11.3.1 IRQ assignment

The IRQ is assigned to pin PTA5 by default .

Chapter 4
Memory Map

4.1 Introduction

This device contains various memories and memory-mapped peripherals which are located in a 4 GB memory space. This chapter describes the memory and peripheral locations within that memory space.

4.2 System memory map

The following table shows the high-level device memory map.

Table 4-1. System memory map

System 32-bit Address Range	Destination Slave	Access
0x0000_0000–0x07FF_FFFF[1]	Program flash and read-only data (Includes exception vectors in first 196 bytes)	Cortex-M0+ core
0x0800_0000–0x0FFF_FFFF	Reserved	—
0x1000_0000– 0x1FFF_FEFF	Reserved	—
0x1FFF_FF00-0x1FFF_FFFF[2]	SRAM_L: Lower SRAM	Cortex-M0+ core
0x2000_0000-0x2000_02FF	SRAM_U: Upper SRAM (bit band region)	Cortex-M0+ core
0x2000_0300–0x21FF_FFFF	Reserved	–
0x2200_0000–0x2200_5FFF	Aliased SRAM_U bit-band region	Cortex-M0+ core
0x2200_6000–0x23FF_FFFF	Reserved	–
0x2400_0000–0x3FFF_FFFF	Bit Manipulation Engine (BME) access to SRAM_U	Cortex-M0+ core
0x4000_0000–0x4007_FFFF	AIPS Peripherals	Cortex-M0+ core
0x4008_0000–0x400F_EFFF	Reserved	–
0x400F_F000–0x400F_FFFF	General purpose input/output (GPIO)	Cortex-M0+ core
0x4010_0000–0x43FF_FFFF	Reserved	–
0x4400_0000–0x5FFF_FFFF	Bit Manipulation Engine (BME) access to AIPS Peripherals for slots 0-127[3]	Cortex-M0+ core
0x6000_0000–0xDFFF_FFFF	Reserved	–

Table continues on the next page...

Table 4-1. System memory map (continued)

System 32-bit Address Range	Destination Slave	Access
0xE000_0000–0xE00F_FFFF	Private Peripherals	Cortex-M0+ core
0xE010_0000–0xEFFF_FFFF	Reserved	–
0xF000_0000–0xF000_0FFF	Reserved	–
0xF000_1000–0xF000_1FFF	Reserved	–
0xF000_2000–0xF000_2FFF	System ROM table[4]	Cortex-M0+ core
0xF000_3000–0xF000_3FFF	Miscellaneous Control Module (MCM)	Cortex-M0+ core
0xF000_4000–0xF7FF_FFFF	Reserved	–
0xF800_0000–0xFFFF_FFFF	IOPORT: FGPIO (single cycle)	Cortex-M0+ core

1. The program flash always begins at 0x0000_0000 but the end of implemented flash varies depending on the amount of flash implemented for a particular device. See Flash memory sizes for details.
2. This range varies depending on SRAM sizes. See SRAM Sizes for details.
3. Includes BME operations to GPIO at slot 15 (based at 0x4000_F000).
4. This device implements a system ROM table which is used to redirect to ARM Cortex M0+ (Flycatcher) ROM table in CoreSight debug system. See System ROM memory map for details.

4.3 Aliased bit-band region

The device supports aliased SRAM_U bit-band region with Cortex M0+ core. A 32-bit write in the alias region has the same result as a read-modify-write operation on the targeted bit in the bit-band region, but with only one cycle time. Aliased bit-band region is much more efficient for bit operation.

Bit 0 of the value written to the alias region determines what value is written to the target bit:

- Writing a value with bit 0 set writes a 1 to the target bit.
- Writing a value with bit 0 clear writes a 0 to the target bit.

A 32-bit read in the alias region returns either:

- a value of 0x0000_0000 to indicate the target bit is clear
- a value of 0x0000_0001 to indicate the target bit is set

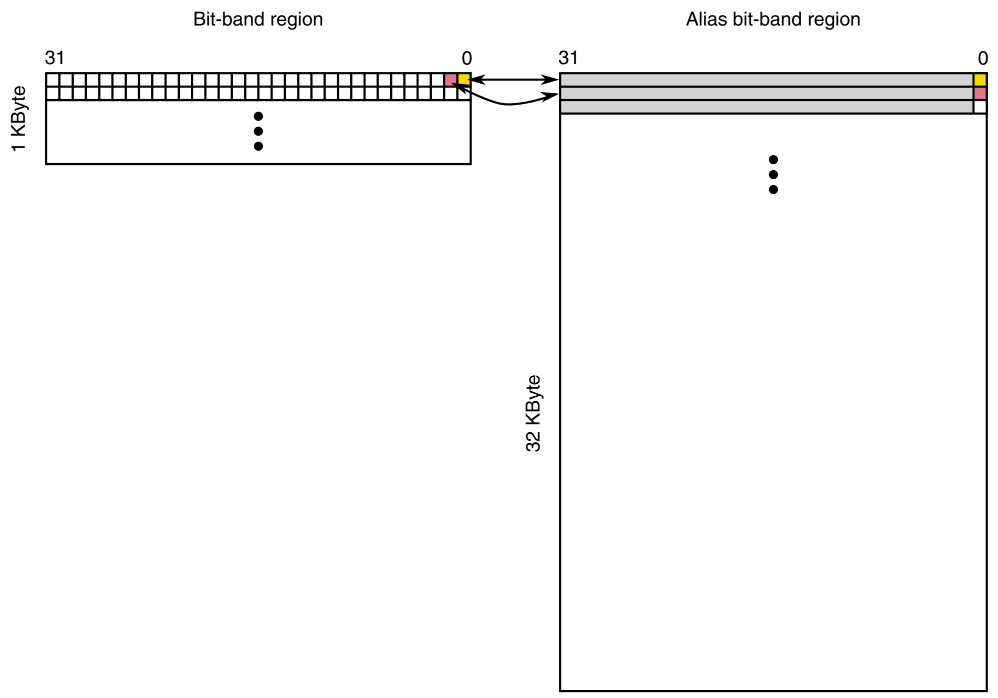


Figure 4-1. Alias bit-band mapping

4.4 Bit Manipulation Engine

The Bit Manipulation Engine (BME) provides hardware support for atomic read-modify-write memory operations to the peripheral and SRAM_U address space. By combining the basic load and store instruction support in the Cortex-M instruction set architecture with the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. See Bit Manipulation Engine (BME) for a detailed description of its functionality.

4.5 System ROM memory map

The system ROM table is optionally required by ARM CoreSight debug infrastructure to discover the components on the chip.

For core configurations like that supported by Cortex-M0+, ARM recommends that a debugger identifies and connects to the debug components using the CoreSight debug infrastructure.

ARM recommends that a debugger follows the flow as shown in the following figure to discover the components in the CoreSight debug infrastructure. In this case, a debugger reads the peripheral and component ID registers for each CoreSight component in the CoreSight system.

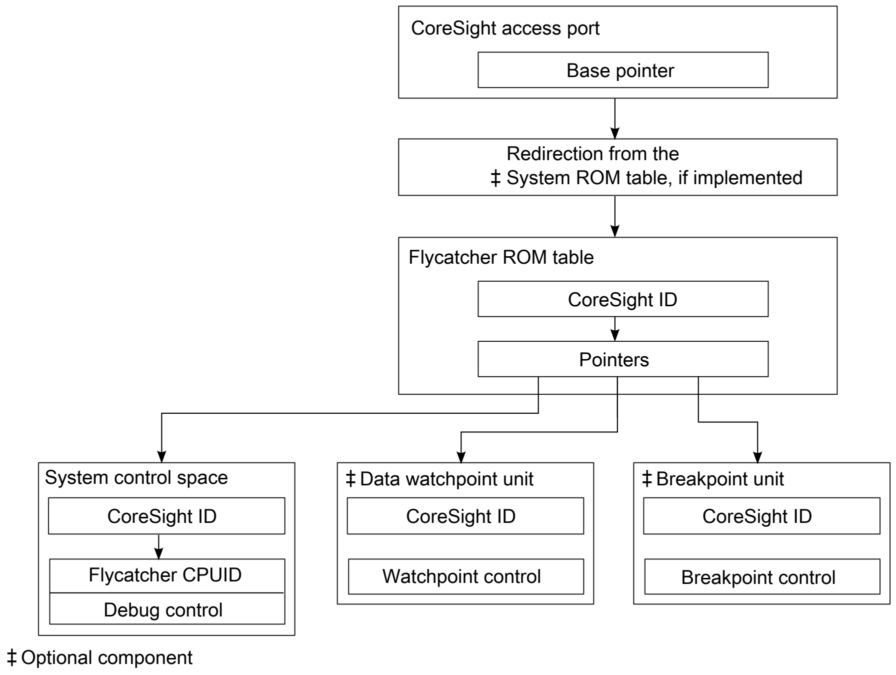


Figure 4-2. CoreSight discovery process

Following table shows the Freescale system ROM table memory map. It includes the ROM entry, peripheral ID and component ID required by ARM CoreSight debug infrastructure.

NOTE

This device contains only standard ARM M0+ core debug components which defined in Flycatcher ROM table. No custom-built debug components are included.

ROM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_2000	Entry (ROM_ENTRY0)	32	R	See section	4.5.1/95
F000_2004	End of Table Marker Register (ROM_TABLEMARK)	32	R	0000_0000h	4.5.2/96
F000_2FCC	System Access Register (ROM_SYSACCESS)	32	R	0000_0001h	4.5.3/96
F000_2FD0	Peripheral ID Register (ROM_PERIPHID4)	32	R	See section	4.5.4/97
F000_2FD4	Peripheral ID Register (ROM_PERIPHID5)	32	R	See section	4.5.4/97

Table continues on the next page...

ROM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_2FD8	Peripheral ID Register (ROM_PERIPHID6)	32	R	See section	4.5.4/97
F000_2FDC	Peripheral ID Register (ROM_PERIPHID7)	32	R	See section	4.5.4/97
F000_2FE0	Peripheral ID Register (ROM_PERIPHID0)	32	R	See section	4.5.4/97
F000_2FE4	Peripheral ID Register (ROM_PERIPHID1)	32	R	See section	4.5.4/97
F000_2FE8	Peripheral ID Register (ROM_PERIPHID2)	32	R	See section	4.5.4/97
F000_2FEC	Peripheral ID Register (ROM_PERIPHID3)	32	R	See section	4.5.4/97
F000_2FF0	Component ID Register (ROM_COMPID0)	32	R	See section	4.5.5/97
F000_2FF4	Component ID Register (ROM_COMPID1)	32	R	See section	4.5.5/97
F000_2FF8	Component ID Register (ROM_COMPID2)	32	R	See section	4.5.5/97
F000_2FFC	Component ID Register (ROM_COMPID3)	32	R	See section	4.5.5/97

4.5.1 Entry (ROM_ENTRY*n*)

The System ROM Table begins with "n" relative 32-bit addresses, one for each debug component present in the device and terminating with an all-zero value signaling the end of the table at the "n+1"-th value.

It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + 0h offset + (4d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENTRY																															
W	▨																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

ROM_ENTRY*n* field descriptions

Field	Description
ENTRY	ENTRY Entry 0 (CM0+ ROM Table) is hardwired to 0xF00F_D003.

4.5.2 End of Table Marker Register (ROM_TABLEMARK)

This register indicates end of table marker. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + 4h offset = F000_2004h

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

ROM_TABLEMARK field descriptions

Field	Description
MARK	Hardwired to 0x0000_0000

4.5.3 System Access Register (ROM_SYSACCESS)

This register indicates system access. It is hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + FCCh offset = F000_2FCCh

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

ROM_SYSACCESS field descriptions

Field	Description
SYSACCESS	Hardwired to 0x0000_0001

4.5.4 Peripheral ID Register (ROM_PERIPHID*n*)

These registers indicate the peripheral IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + FD0h offset + (4d × i), where i=0d to 7d

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

* Notes:
- x = Undefined at reset.

ROM_PERIPHID*n* field descriptions

Field	Description
PERIPHID	Peripheral ID1 is hardwired to 0x0000_00E0; ID2 to 0x0000_0008; and all the others to 0x0000_0000.

4.5.5 Component ID Register (ROM_COMPID*n*)

These registers indicate the component IDs. They are hardwired to specific values used during the auto-discovery process by an external debug agent.

Address: F000_2000h base + FF0h offset + (4d × i), where i=0d to 3d

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

* Notes:
- x = Undefined at reset.

ROM_COMPID*n* field descriptions

Field	Description
COMPID	Component ID Component ID0 is hardwired to 0x0000_000D; ID1 to 0x0000_0010; ID2 to 0x0000_0005; ID3 to 0x0000_00B1.

4.6 Peripheral bridge (AIPS-Lite) memory map

The Peripheral Bridge memory map is accessible via one slave port on the crossbar in the 0x4000_0000–0x400F_FFFF region. The device implements one peripheral bridge that defines a 1024 KB address space.

The three regions associated with this space are:

- A 128 KB region, partitioned as 32 spaces, each 4 KB in size and reserved for on-platform peripheral devices. The AIPS controller generates unique module enables for all 32 spaces.
- A 384 KB region, partitioned as 96 spaces, each 4 KB in size and reserved for off-platform modules. The AIPS controller generates unique module enables for all 96 spaces.
- The last slot is a 4 KB region beginning at 0x400F_F000 for accessing the GPIO module. The GPIO slot (slot 128) is an alias of slot 15. This block is also directly interfaced to the core and provides direct access without incurring wait states associated with accesses via the AIPS controller.

Modules that are disabled via their clock gate control bits in the SIM registers disable the associated AIPS slots. Access to any address within an unimplemented or disabled peripheral bridge slot results in a transfer error termination.

For programming model accesses via the peripheral bridges, there is generally only a small range within the 4 KB slots that is implemented. Accessing an address that is not implemented in the peripheral results in a transfer error termination.

4.6.1 Read-after-write sequence and required serialization of memory operations

In some situations, a write to a peripheral must be completed fully before a subsequent action can occur. Examples of such situations include:

- Exiting an interrupt service routine (ISR)
- Changing a mode
- Configuring a function

In these situations, the application software must perform a read-after-write sequence to guarantee the required serialization of the memory operations:

1. Write the peripheral register.
2. Read the written peripheral register to verify the write.
3. Continue with subsequent operations.

4.6.2 Peripheral Bridge (AIPS-Lite) Memory Map

NOTE

- Slots 0-95 and 128 are 32-bit data width modules, with the exception that slots 49,82 are 8-bit data width modules (IRQ, WDOG)
- Slots 96-127 are 8-bit data width modules.

Table 4-21. Peripheral bridge 0 slot assignments

System 32-bit base address	Slot number	Module
0x4000_0000	0	—
0x4000_1000	1	—
0x4000_2000	2	—
0x4000_3000	3	—
0x4000_4000	4	—
0x4000_5000	5	—
0x4000_6000	6	—
0x4000_7000	7	—
0x4000_8000	8	—
0x4000_9000	9	—
0x4000_A000	10	—
0x4000_B000	11	—
0x4000_C000	12	—
0x4000_D000	13	—
0x4000_E000	14	—
0x4000_F000	15	GPIO controller (aliased to 0x400F_F000)
0x4001_0000	16	—
0x4001_1000	17	—
0x4001_2000	18	—
0x4001_3000	19	—
0x4001_4000	20	—
0x4001_5000	21	—
0x4001_6000	22	—
0x4001_7000	23	—
0x4001_8000	24	—
0x4001_9000	25	—
0x4001_A000	26	—
0x4001_B000	27	—
0x4001_C000	28	—

Table continues on the next page...

Table 4-21. Peripheral bridge 0 slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4001_D000	29	—
0x4001_E000	30	—
0x4001_F000	31	—
0x4002_0000	32	Flash memory (FTMRE)
0x4002_1000	33	—
0x4002_2000	34	—
0x4002_3000	35	—
0x4002_4000	36	—
0x4002_5000	37	—
0x4002_6000	38	—
0x4002_7000	39	—
0x4002_8000	40	—
0x4002_9000	41	—
0x4002_A000	42	—
0x4002_B000	43	—
0x4002_C000	44	—
0x4002_D000	45	—
0x4002_E000	46	—
0x4002_F000	47	—
0x4003_0000	48	—
0x4003_1000	49	IRQ controller (IRQ)
0x4003_2000	50	Cyclic Redundancy Check (CRC)
0x4003_3000	51	Pulse width timer (PWT)
0x4003_4000	52	—
0x4003_5000	53	—
0x4003_6000	54	—
0x4003_7000	55	Periodic interrupt timers (PIT)
0x4003_8000	56	Flex timer 0 (FTM0)
0x4003_9000	57	—
0x4003_A000	58	Flex timer 2 (FTM2)
0x4003_B000	59	Analog-to-digital converter (ADC)
0x4003_C000	60	—
0x4003_D000	61	Real time clock (RTC)
0x4003_E000	62	—
0x4003_F000	63	—
0x4004_0000	64	—
0x4004_1000	65	—
0x4004_2000	66	—
0x4004_3000	67	—

Table continues on the next page...

Table 4-21. Peripheral bridge 0 slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4004_4000	68	—
0x4004_5000	69	—
0x4004_6000	70	—
0x4004_7000	71	—
0x4004_8000	72	System integration module (SIM)
0x4004_9000	73	Port controller
0x4004_A000	74	—
0x4004_B000	75	—
0x4004_C000	76	—
0x4004_D000	77	—
0x4004_E000	78	—
0x4004_F000	79	—
0x4005_0000	80	—
0x4005_1000	81	—
0x4005_2000	82	Watchdog (WDOG)
0x4005_3000	83	—
0x4005_4000	84	—
0x4005_5000	85	—
0x4005_6000	86	—
0x4005_7000	87	—
0x4005_8000	88	—
0x4005_9000	89	—
0x4005_A000	90	—
0x4005_B000	91	—
0x4005_C000	92	—
0x4005_D000	93	—
0x4005_E000	94	—
0x4005_F000	95	—
0x4006_0000	96	—
0x4006_1000	97	—
0x4006_2000	98	—
0x4006_3000	99	—
0x4006_4000	100	Internal clock source (ICS)
0x4006_5000	101	System oscillator (OSC)
0x4006_6000	102	I^{2}C0
0x4006_7000	103	—
0x4006_8000	104	—
0x4006_9000	105	—
0x4006_A000	106	Universal asynchronous receiver/transmitter 0 (UART0)

Table continues on the next page...

Table 4-21. Peripheral bridge 0 slot assignments (continued)

System 32-bit base address	Slot number	Module
0x4006_B000	107	—
0x4006_C000	108	—
0x4006_D000	109	—
0x4006_E000	110	—
0x4006_F000	111	—
0x4007_0000	112	—
0x4007_1000	113	—
0x4007_2000	114	—
0x4007_3000	115	Analog comparator 0 (ACMP0)
0x4007_4000	116	Analog comparator 1 (ACMP1)
0x4007_5000	117	—
0x4007_6000	118	Serial peripheral interface 0 (SPI0)
0x4007_7000	119	—
0x4007_8000	120	—
0x4007_9000	121	Keyboard interrupt 0 (KBI0)
0x4007_A000	122	Keyboard interrupt 1 (KBI1)
0x4007_B000	123	—
0x4007_C000	124	—
0x4007_D000	125	Power management controller (PMC)
0x4007_E000	126	—
0x4007_F000	127	—
0x400F_F000	128	GPIO controller

4.7 Private Peripheral Bus (PPB) memory map

The PPB is part of the defined ARM bus architecture and provides access to select processor-local modules. These resources are only accessible from the core; other system masters do not have access to them.

Table 4-22. PPB memory map

System 32-bit Address Range	Resource	Additional Range Detail	Resource
0xE000_0000–0xE000_DFFF	Reserved		
0xE000_E000–0xE000_EFFF	System Control Space (SCS)	0xE000_E000–0xE000_E00F	Reserved
		0xE000_E010–0xE000_E0FF	SysTick
		0xE000_E100–0xE000_ECFF	NVIC
		0xE000_ED00–0xE000_ED8F	System Control Block

Table continues on the next page...

Table 4-22. PPB memory map (continued)

System 32-bit Address Range	Resource	Additional Range Detail	Resource
		0xE000_ED90–0xE000_EDEF	Reserved
		0xE000_EDF0–0xE000_EEFF	Debug
		0xE000_EF00–0xE000_EFFF	Reserved
0xE000_F000–0xE00F_EFFF	Reserved		
0xE00F_F000–0xE00F_FFFF	Core ROM Space (CRS)		

Chapter 5
Clock Distribution

5.1 Introduction

This chapter presents the clock architecture for the device, the overview of the clocks and includes a terminology section.

The Cortex M0+ resides within a synchronous core platform, where the processor and bus masters, flash and peripherals clocks can be configured independently.

The ICS module will be used for main system clock generation. The ICS module controls which clock sources (internal references, external crystals or external clock signals) generate the source of the system clocks.

5.2 Programming model

The selection and multiplexing of system clock sources is controlled and programmed via the ICS module. The setting of clock dividers and module clock gating for the system are programmed via the SIM module. See those sections for detailed register and bit descriptions.

5.3 High-level device clocking diagram

This device contains following on-chip clock sources:

- Internal Clock Source (ICS) module: The main clock source generator providing bus clock and other reference clocks to peripherals

- System Oscillator (OSC) module: The system oscillator providing reference clock to internal clock source (ICS), the real-time clock counter clock module (RTC), and other MCU sub-systems
- Low-Power Oscillator (LPO) module: The on-chip low-power oscillator providing 1 kHz reference clock to RTC and Watchdog (WDOG)

Figure 5-1 shows how clocks from the ICS and OSC modules are distributed to the microcontroller's other function units. Some modules in the microcontroller have selectable clock input.

The following registers of the system oscillator, ICS, and SIM modules control the multiplexers, dividers, and clock gates shown in the figure:

Table 5-1. Registers controlling multiplexers, dividers, and clock gate

	OSC	ICS	SIM
Multiplexers	OSC_CR	ICS_C1	SIM_SOPT
Dividers	—	ICS_C2	SIM_CLKDIV
Clock gates	OSC_CR	ICS_C1	SIM_SCGC

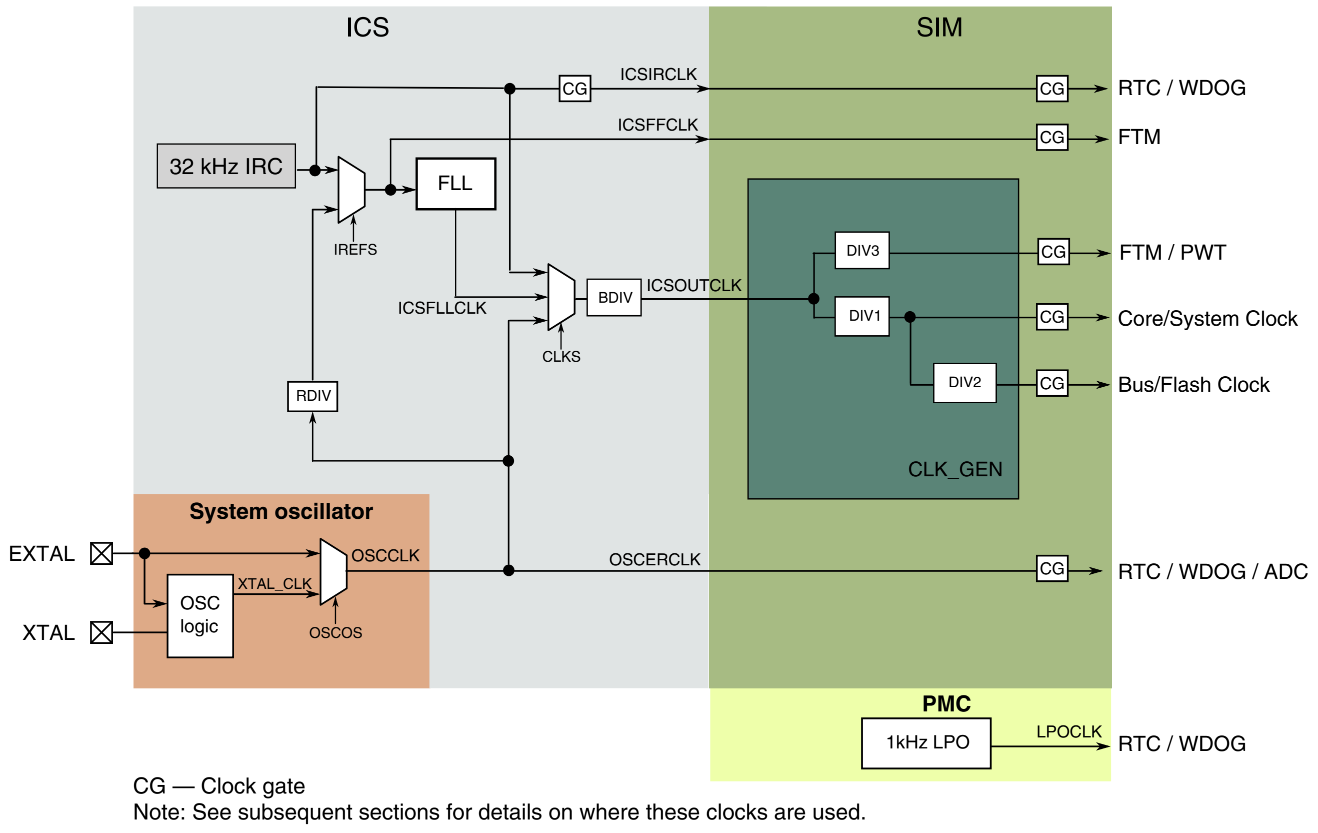


Figure 5-1. Clocking diagram

5.4 Clock definitions

The following table describes the clocks in Figure 5-1.

Table 5-2. Clock definitions

Clock name	Description
Core clock	ICSOUTCLK divided by DIV1, clocks the ARM Cortex-M0+ core. It's the CPU HCLK
Platform clock	ICSOUTCLK divided by DIV1, clocks the crossbar switch and NVIC, It's the free-running FCLK
System clock	ICSOUTCLK divided by DIV1, clocks the bus masters directly
Bus clock	System clock divided by DIV2, clocks the bus slaves and peripherals
Flash clock	System clock divided by DIV2, clocks the Flash memory module. It is same as Bus clock in this device
Timer clock	ICSOUTCLK divided by DIV3, clocks the FTM and PWT modules
Debug clock	Debug logic clock. On this device it is derived from platform clock
SWD clock	DAP interface clock. SWD clock is typically driven by an external debugger and completely asynchronous to Core clock and platform clock
ICSIRCLK	ICS output of the internal 32 kHz IRC reference clock. ICSIRCLK can be selected as the clock source of RTC or WDOG modules
ICSOUTCLK	ICS output of either IRC, ICSFLLCLK or ICS's external reference clock that sources the core, system, bus, and flash clock
ICSFLLCLK	Output of the FLL, FLL locks the frequency to 1280 times the internal or external reference frequency
ICSFFCLK	ICS output of the fixed frequency clock. ICSFFCLK can be selected as clock source for the FTM modules. The frequency of the ICSFFCLK is determined by the setting of the ICS
OSCCLK	System oscillator output of the internal oscillator or sourced directly from EXTAL. Used as ICS external reference clock
OSCERCLK	System oscillator output sourced from OSCCLK that can be selected as the clock source of RTC, WDOG or ADC modules
LPOCLK	PMC 1 kHz output, The LPOCLK can be selected as the clock source to the RTC or WDOG modules

5.4.1 Device clock summary

The following table provides more information regarding the on-chip clocks.

Table 5-3. Clock summary

Clock name	Run mode frequency	Clock source	Clock is disabled when…
Core clock	Up to 48 MHz	ICSOUTCLK clock divider	In Wait and Stop modes

Table continues on the next page...

Table 5-3. Clock summary (continued)

Clock name	Run mode frequency	Clock source	Clock is disabled when…
Platform clock	Up to 48 MHz	ICSOUTCLK clock divider	In Stop mode
System clock	Up to 48 MHz	ICSOUTCLK clock divider	In Stop mode
Timer clock	Up to 48 MHz	ICSOUTCLK clock divider	In Stop mode
Bus clock	Up to 24 MHz	ICSOUTCLK clock divider	In Stop mode
Debug clock	Up to 24 MHz	Derive from Platform clock	Debug not enabled
SWD clock	Up to 24MHz	SWD_CLK pin	Input from external clock, so will not be disabled.
Flash clock	Up to 24 MHz	ICSOUTCLK clock divider	In Stop mode
Internal reference (ICSIRCLK)	31.25–39.0625 kHz IRC	IRC	ICS_C1[IRCLKEN]=0, or In Stop mode and ICS_C1[IREFSTEN]=0
External reference (OSCERCLK)	DC up to 48 MHz (bypass), 31.25–39.0625 kHz or 4–24 MHz (crystal)	System OSC	OSC_CR[OSCEN]=0, or In Stop mode and OSC_CR[OSCSTEN]=0
FLL out clock (ICSFLLCLK)	40-50 MHz	System OSC or IRC	In Stop mode, or FLL not enabled
ICS Fixed Frequency clock (ICSFFCLK)	31.25–39.0625 kHz	System OSC or IRC	In Stop mode
LPOCLK	1 kHz	PMC	Available in all power modes

5.4.2 Clock distribution

The following figure shows a simplified clock distribution diagram

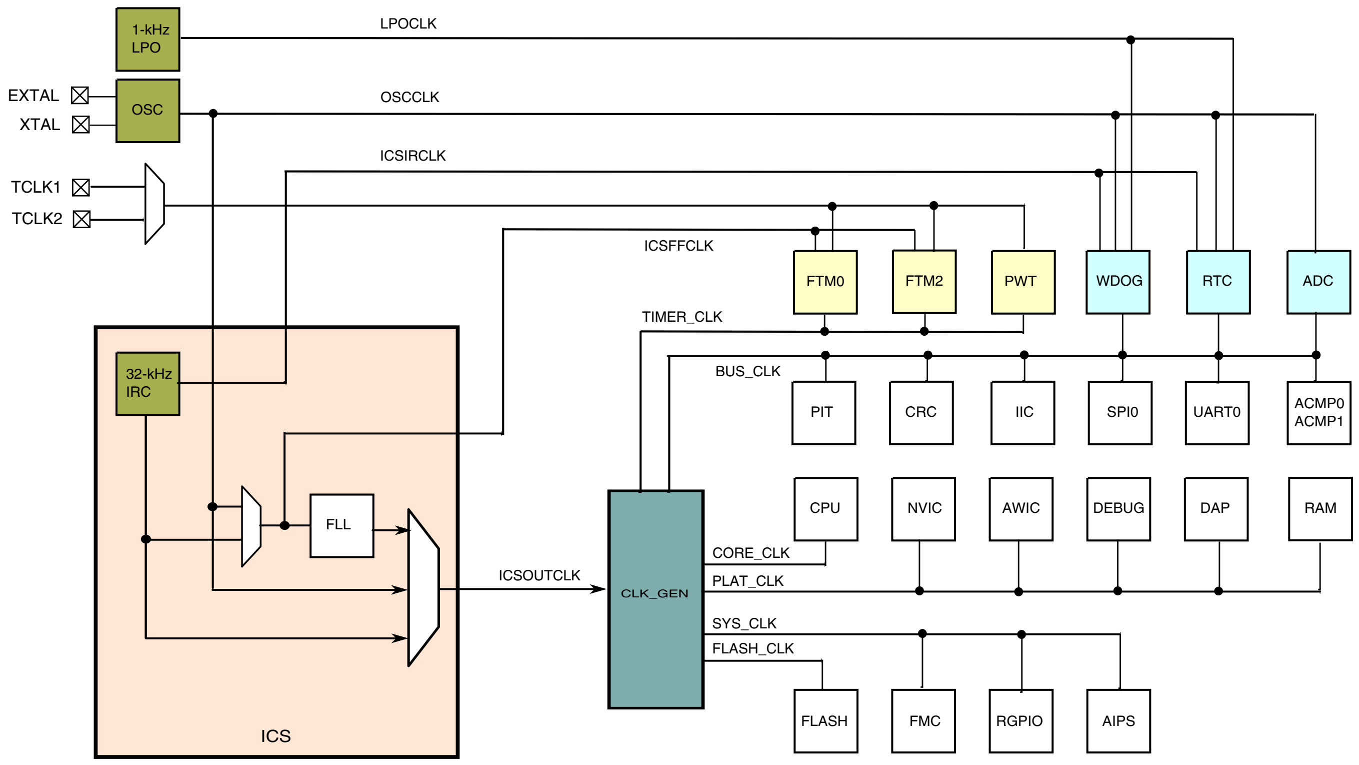


Figure 5-2. High-level clock distribution diagram

NOTE

Clock divide and gating are not shown in clock distribution diagram.

5.5 Internal clocking sources

The internal clock sources on this device are as following:

- On-chip RC oscillator range of 31.25–39.0625 kHz as the reference of FLL input.
- On-chip internal 1 kHz oscillator as the low-frequency low-power source for RTC and WDOG according to specific use case requirement.

The following table shows the frequency availability of this device:

Table 5-4. ICS bus frequency availability with internal reference

Reference		ICSOUTCLK
FEI (high range)	BDIV = 0	40 MHz ~ 50 MHz[1]
	BDIV = 1	20 MHz ~ 25 MHz
	BDIV = 2	10 MHz ~ 12.5 MHz
	BDIV = 4	5 MHz ~ 6.25 MHz

Table continues on the next page...

Table 5-4. ICS bus frequency availability with internal reference (continued)

Reference		ICSOUTCLK
	BDIV = 8	2.5 MHz ~ 3.125 MHz
	BDIV = 16	1.25 MHz ~ 1.5625 MHz
	BDIV = 32	625 kHz ~ 781.25 kHz
	BDIV = 64	312.5 kHz ~ 390.625 kHz
	BDIV = 128	156.25 kHz ~ 195.3125 kHz

1. Carefully configure SIM_CLKDIV and BDIV to avoid any clock frequency higher than 48 MHz.

5.6 External clock sources

This device supports the following two external clock sources:

- External square wave input clock up to DC-48 MHz.
- External crystal oscillator or resonator:
  - Low-range: 31.25–39.0625 kHz
  - High-range: 4–24 MHz

NOTE

The external square wave input clock is only used when the OSC module is working under external clock mode (bypass). With the external square wave clock source, the user can use FBE mode with FLL disabled to achieve lower power consumption or precise clock source.

The following table shows the frequency availability of this device when sourcing from OSC clock. OSC external clock mode is not shown.

Table 5-5. OSC frequency availability

ICS configuration	External reference	RDIV
FBE	31.25 kHz ~ 39.0625 kHz	–
	4 MHz ~ 24 MHz	–
FEE[1]	31.25 kHz ~ 39.0625 kHz	RDIV = 1
	62.5 kHz ~ 78.125 kHz	RDIV = 2
	125 kHz ~ 56.25 kHz	RDIV = 4
	250 kHz ~ 312.5 kHz	RDIV = 8
	500 kHz ~ 625 kHz	RDIV = 16
	1 MHz ~ 1.25 MHz	RDIV = 32
	2 MHz ~ 2.5 MHz	RDIV = 64
	4 MHz ~ 5 MHz	RDIV = 128

Table continues on the next page...

Table 5-5. OSC frequency availability (continued)

ICS configuration	External reference	RDIV
	8 MHz ~ 10 MHz	RDIV = 256
	16 MHz ~ 20 MHz	RDIV = 512

1. In FEE mode, FLL output frequency = OSC/RDIV *1280. Select the OSC and RDIV carefully to keep the FLL output frequency within the limits.

5.7 Clock gating

The clock to each module can be individually gated on and off using the System Clock Gating Control Register (SIM_SCGC). Prior to initializing a module, set the corresponding bit in System Clock Gating Control Register (SIM_SCGC) to enable the clock. Before turning off the clock, make sure to disable the module.

Any bus access to a peripheral that has its clock disabled generates an error termination.

5.8 Module clocks

The following table summarizes the clocks associated with each module.

Table 5-6. Module clocks

Module	Bus interface clock	Internal clocks	I/O interface clocks
Core modules			
ARM Cortex-M0+ core	Platform clock	Core clock	—
NVIC	Platform clock	—	—
DAP	Platform clock	—	SWD_CLK
System modules			
Port control	Bus clock	—	—
Crossbar Switch	Platform clock	—	—
Peripheral bridges	System clock	Bus clock	—
PMC, SIM	Bus clock	LPOCLK	—
MCM	Platform clock	—	—
CRC	Bus clock	—	—
Watchdog timer	Bus clock	Bus clock LPOCLK ICSIRCLK OSCERCLK	—
Clocks			

Table continues on the next page...

Table 5-6. Module clocks (continued)

Module	Bus interface clock	Internal clocks	I/O interface clocks
ICS	Bus clock	ICSOUTCLK ICSFLLCLK ICSIRCLK OSCERCLK	—
OSC	Bus clock	OSCERCLK	—
Memory and memory interfaces			
Flash Controller	System clock	—	—
Flash memory	Flash clock	—	—
SRAM	Platform clock	—	—
Analog			
ADC	Bus clock	Bus clock OSCERCLK ADACK	—
ACMP0	Bus clock	—	—
ACMP1	Bus clock	—	—
Timers			
PIT	Bus clock	—	—
PWT	Timer clock	—	TCLK1, TCLK2
FTM0	Timer clock	Timer clock ICSFFCLK	TCLK1, TCLK2
FTM2	Timer clock	Timer clock ICSFFCLK	TCLK1, TCLK2
RTC	Bus clock	Bus clock LPOCLK ICSIRCLK OSCERCLK	RTC_CLKOUT
Communication interfaces			
SPI0	Bus clock	—	SPI0_SCK
I^{2}C0	Bus clock	—	I2C0_SCL
UART0	Bus clock	—	—
Human-machine interfaces			
GPIO	System clock	—	—
KBI0	Bus clock	—	—
KBI1	Bus clock	—	—

5.8.1 FTM and PWT clocking

The counters for the FTM and PWT modules have a selectable clock as shown in the following figure. TCLK1 and TCLK2 are optional external clock inputs for the timers.

NOTE

If TCLK1 or TCLK2 are selected as the counter clock for FTMs or PWT, the on-chip timer clock (TIMER_CLK) is still required to synchronize the counter results. And the on-chip timer clock (TIMER_CLK) must be at least 4x faster than the external clock from TCLK1 or TCLK2.

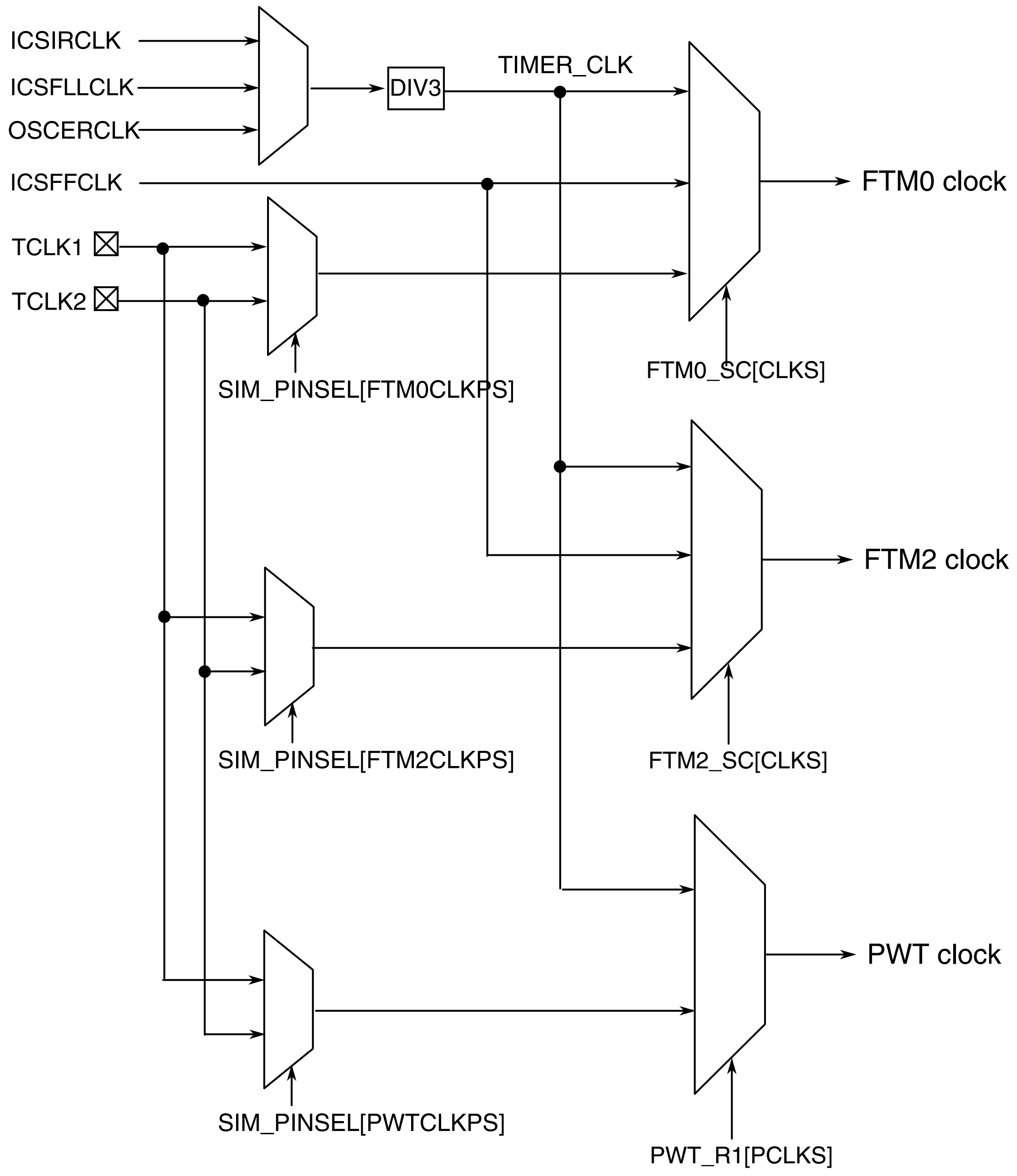


Figure 5-3. FTM and PWT clock generation

Chapter 6
Reset and Boot

6.1 Introduction

The following reset sources are supported in this MCU:

Table 6-1. Reset sources

Reset sources	Description
POR reset	• Power-on reset (POR)
System resets	• External pin reset (PIN) • Low-voltage detect (LVD) • Watchdog (WDOG) timer • ICS loss of clock (LOC) reset • Stop mode acknowledge error (SACKERR) • Software reset (SW) • Lockup reset (LOCKUP) • MDM DAP system reset

Each of the system reset sources has an associated bit in the System Reset Status and ID Register (SIM_SRSID).

The MCU can exit and reset in functional mode where the CPU is executing code (default) or the CPU is in a debug halted state. There are several boot options that can be configured. See Boot for more details.

6.2 Reset

This section discusses basic reset mechanisms and sources. Some modules that cause resets can be configured to cause interrupts instead. Consult the individual peripheral chapters for more information.

6.2.1 Power-on reset (POR)

When power is initially applied to the MCU or when the supply voltage drops below the power-on reset voltage level (V_{POR}), the POR circuit causes a POR reset condition.

As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply has risen above the LVD low threshold (V_{LVDL}). POR and LVD fields of the System Reset Status and ID Register (SIM_SRSID). (SIM_SRSID[POR] and SIM_SRSID[LVD]) are set following a POR.

6.2.2 System reset sources

Resetting the MCU provides a way to start processing from a known set of initial conditions. System reset begins with the on-chip regulator in full regulation and system clocking generation from an internal reference. When the processor exits reset, it performs the following:

- Reads the start SP (SP_main) from vector-table offset 0
- Reads the start program counter (PC) from vector-table offset 4
- The Link Register (LR) is set to 0xFFFF_FFFF.

The on-chip peripheral modules are disabled and the non-analog I/O pins are initially configured as disabled (except that the SWD_DIO/SWD_CLK, $\overline{\text{NMI}}$ and $\overline{\text{RESET}}$ pins could be enabled after system reset according to the System Options Register (SIM_SOPT) setting). The pins with analog functions assigned to them default to their analog function after reset.

6.2.2.1 External pin reset ($\overline{\text{RESET}}$)

This pin has an internal pullup resistor. Asserting $\overline{\text{RESET}}$ wakes the device from any mode.

After POR reset, the PTA5 pin functions as $\overline{\text{RESET}}$. SIM_SOPT[RSTPE] must be programmed to enable the other functions. When this field is clear, this pin can function as PTA5 or other alternative functions.

6.2.2.1.1 Reset pin filter

The $\overline{\text{RESET}}$/IRQ pin filter supports filtering from both the 1 kHz LPO clock and the bus clock. It can be used as a simple low-pass filter to filter any glitch that is introduced from the pin of $\overline{\text{RESET}}$/IRQ.

The glitch width threshold can be adjusted easily by setting Port Filter Register (PORT_IOFLT) between 1~4096 BUSCLKs (or 1~128 LPOCLKs). This configurable glitch filter can replace an on-board external analog filter, and greatly improve the EMC performance. Setting Port Filter Register (PORT_IOFLT) can configure the filter of the whole port.

6.2.2.2 Low-voltage detect (LVD)

This device includes a system to protect against low-voltage conditions in order to protect memory contents and control MCU system states during supply voltage variations. This system consists of a power-on reset (POR) circuit, and an LVD circuit with a user selectable trip voltage, either high (V_{LVDH}) or low (V_{LVDL}).

The LVD circuit is enabled when PMC_SPMSC1[LVDE] is set and the trip voltage is selected by PMC_SPMSC2[LVDV]. The LVD is disabled upon entering the stop modes unless PMC_SPMSC1[LVDSE] is set or in Serial Wire Debug (SWD) mode. If PMC_SPMSC1[LVDSE] and PMC_SPMSC1[LVDE] are both set, the current consumption will be higher in Stop mode with the LVD enabled.

6.2.2.3 Watchdog timer

The watchdog timer (WDOG) monitors the operation of the system by expecting periodic communication from the software. This communication is generally known as servicing (or refreshing) the watchdog. If this periodic refreshing does not occur, the watchdog issues a system reset. The WDOG reset causes SIM_SRSID[WDOG] to set.

6.2.2.4 ICS loss-of-clock (LOC)

The ICS on this chip supports external reference clock monitor with reset capability.

In FBE, or FEE modes, if 1 is written to ICS_C4[CME], the clock monitor is enabled. If the external reference falls below a certain frequency, such as f_{loc_high} or f_{loc_low} depending on OSC_CR[RANGE], the MCU will reset. SIM_SRSID[LOC] will be set to indicate the error.

In FBILP mode, the FLL is not on, so the external reference clock monitor will not function even if 1 is written to ICS_C4[CME].

External reference clock monitor uses FLL as the internal reference clock. The FLL must be functional before ICS_C4[CME] is set.

6.2.2.5 Stop mode acknowledge error (SACKERR)

This reset is generated if the core attempts to enter Stop mode, but not all modules acknowledge Stop mode within 1025 cycles of the 1 kHz LPO clock.

A module might not acknowledge the entry to Stop mode if an error condition occurs. The error can be caused by a failure of an external clock input to a module.

6.2.2.6 Software reset (SW)

The SYSRESETREQ field in the NVIC application interrupt and reset control register can be set to force a software reset on the device. (See ARM's NVIC documentation for the full description of the register fields, especially the VECTKEY field requirements.) Setting SYSRESETREQ generates a software reset request. This reset forces a system reset of all major components except for the debug module.

6.2.2.7 Lockup reset (LOCKUP)

The LOCKUP gives immediate indication of seriously errant kernel software. This is the result of the core being locked because of an unrecoverable exception following the activation of the processor’s built in system state protection hardware.

The LOCKUP condition causes a system reset and also causes SIM_SRSID[LOCKUP] to set.

6.2.2.8 MDM-AP system reset request

Set the System Reset Request field in the MDM-AP Control register to initiate a system reset. This is the primary method for resets via the SWD interface. The system reset is held until this bit is cleared.

Set the Core Hold Reset field in the MDM-AP Control register to hold the core in reset as the rest of the chip comes out of system reset.

6.2.3 MCU resets

A variety of resets are generated by the MCU to reset different modules.

6.2.3.1 POR Only

The POR Only reset asserts on the POR reset source only. It resets the PMC and RTC.

The POR Only reset also causes all other reset types to occur.

6.2.3.2 Chip POR

The Chip POR asserts on POR and LVD reset sources. It resets the Reset Pin Filter registers and parts of the SIM and ICS.

The Chip POR also causes the Chip Reset (including Early Chip Reset) to occur.

6.2.3.3 Early Chip Reset

The Early Chip Reset asserts on all reset sources. It resets only the flash memory module and ARM platform. It negates before flash memory initialization begins ("earlier" than when the Chip Reset negates).

6.2.3.4 Chip Reset

Chip Reset asserts on all reset sources and only negates after the $\overline{\text{RESET}}$ pin has also negated. It resets the remaining modules (the modules not reset by other reset types).

6.3 Boot

This section describes the boot sequence, including sources and options.

Some configuration information such as clock trim values stored in factory programmed flash locations is auto-loaded.

6.3.1 Boot sources

The CM0+ core adds support for a programmable Vector Table Offset Register (VTOR[1]) to relocate the exception vector table. This device supports booting from internal flash and RAM.

1. VTOR: refer to Vector Table Offset Register in the ARMv6-M Architecture Reference Manual.

This device supports booting from internal flash with the reset vectors located at addresses 0x0 (initial SP_main), 0x4 (initial PC), and RAM with relocating the exception vector table to RAM.

6.3.2 Boot sequence

At power-up, the on-chip regulator holds the system in a POR state until the input supply is above the POR threshold. The system continues to be held in this static state until the internally regulated supplies have reached a safe operating voltage as determined by the LVD. The Reset Controller logic then controls a sequence to exit reset.

1. A system reset is held on internal logic, the $\overline{\text{RESET}}$ pin is driven out low (about 4.2 µs), and the ICS is enabled in its default clocking mode.
2. The $\overline{\text{RESET}}$ pin is released. If $\overline{\text{RESET}}$ pin continues to be asserted (an indication of a slow rise time on the $\overline{\text{RESET}}$ pin or external drive in low), the system continues to be held in reset. Once the $\overline{\text{RESET}}$ pin is detected high, the core clock is enabled and the system is released from reset.
3. The NVM starts internal initialization. Flash Controller is released from reset and begins initialization operations while the core is still halted before the flash initialization completes.
4. When the flash Initialization completes(16 µs) , the core sets up the stack, program counter (PC), and link register (LR). The processor reads the start SP (SP_main) from vector-table offset 0. The core reads the start PC from vector-table offset 4. LR is set to 0xFFFF_FFFF. The CPU begins execution at the PC location.

Subsequent system resets follow this same reset flow.

Chapter 7
Power Management

7.1 Introduction

This chapter describes the various chip power modes and functionality of the individual modules in these modes.

7.2 Power modes

The power management controller (PMC) provides the user with multiple power options. The different modes of operation are supported to allow the user to optimize power consumption for the level of functionality needed.

The device supports Run, Wait, and Stop modes which are easy to use for customers both from different power consumption level and functional requirement. I/O states are held in all the modes.

- Run mode—CPU clocks can be run at full speed and the internal supply is fully regulated.
- Wait mode—CPU shuts down to conserve power; system clocks and bus clock are running and full regulation is maintained.
- Stop mode—LVD optional enabled, and voltage regulator is in standby.

The three modes of operation are Run, Wait, and Stop. The WFI instruction invokes both Wait and Stop modes for the chip.

Table 7-1. Chip power modes

Power mode	Description	Core mode	Normal recover method
Normal RUN	Allows maximum performance of chip. Default mode out of reset; on-chip voltage regulator is on.	Run	—

Table continues on the next page...

Table 7-1. Chip power modes (continued)

Power mode	Description	Core mode	Normal recover method
Normal Wait via WFI	Allows peripherals to function while the core is in Sleep mode, reducing power. NVIC remains sensitive to interrupts; peripherals continue to be clocked.	Sleep	Interrupt
Normal Stop via WFI	Places chip in static state. Lowest power mode that retains all registers while optionally maintaining LVD protection. NVIC is disabled; AWIC is used to wake up from interrupt; peripheral clocks are stopped.	Sleep Deep	Interrupt

7.3 Entering and exiting power modes

The WFI instruction invokes Wait and Stop modes for the chip. The processor exits the low-power mode via an interrupt.

NOTE

The WFE instruction can have the side effect of entering a low-power mode, but that is not its intended usage. See ARM documentation for more on the WFE instruction.

7.4 Module operation in low-power modes

The following table illustrates the functionality of each module while the chip is in each of the low-power modes. The standard behavior is shown with some exceptions.

Table 7-2. Module operation in low-power modes

Modules	Run	Wait	Stop
Core modules			
CPU	On	Standby	Standby
NVIC	On	On	Standby
System modules			
PMC	Full regulation	Full regulation	Loose regulation
WDOG	On	On	Optional on
LVD	On	On	Optional on
CRC	On	On	Standby
Clock			
ICS	On	On	Optional on
OSC	On	On	Optional on
LPO	On	On	Always on

Table continues on the next page...

Table 7-2. Module operation in low-power modes (continued)

Modules	Run	Wait	Stop
Memory			
Flash	On	On	Standby
RAM	On	Standby[1]	Standby
Timer			
FTM	On	On	Standby
PIT	On	On	Standby
PWT	On	On	Standby
RTC	On	On	Optional on
Analog			
ADC	On	On	Optional on
ACMP	On	On	Optional on
Communication interfaces			
UART	On	On	Standby[2]
SPI	On	On	Standby[3]
IIC	On	On	Standby[4]
Human-machine interfaces			
KBI	On	On	Standby[5]
IRQ	On	On	Standby[5]
I/O	On	On	State held

1. SRAM enable signal disables internal clock signal and masks the address and data inputs when held low, RAM clock at chip can be active in Wait mode.
2. Supports wake-up on edge in Stop mode
3. Supports slave mode receive and wake-up in Stop mode
4. Supports address match wake-up in Stop mode
5. Supports pin interrupt wake-up in Stop mode

Chapter 8
Security

8.1 Introduction

This device implements security based on the mode selected from the flash module. The following sections provide an overview of flash security and details of the effects of security on non-flash modules.

8.2 Flash security

The flash module provides security information to the MCU based on the state held by the FTMRE_FSEC[SEC]. The MCU, in turn, confirms the security request and limits access to flash resources. During reset, the flash module initializes the Flash Security Register (FTMRE_FSEC) using data read from the security byte of the flash configuration field.

NOTE

The security features apply only to external accesses: CPU accesses to the flash are not affected by the status of Flash Security Register (FTMRE_FSEC).

In the unsecured state all flash commands are available on the programming interfaces either from the debug port (SWD) or user code execution. When the flash is secured (FTMRE_FSEC[SEC] = 00, 01, or 11), the programmer interfaces are only allowed to launch mass erase operations. Additionally, in this mode, the debug port has no access to memory locations.

8.3 Security interactions with other modules

The flash security settings are used by the system to determine what resources are available. The following sections describe the interactions between modules and the flash security settings or the impact that the flash security has on non-flash modules.

8.3.1 Security interactions with debug

When flash security is active, the SWD port cannot access the memory resources of the MCU.

Although most debug functions are disabled, the debugger can write to the Flash Mass Erase in Progress field of the MDM-AP Control Register to trigger a mass erase (Erase All Blocks) command. A mass erase via the debugger is allowed even when some memory locations are protected.

Chapter 9
Debug

9.1 Introduction

This device's debug is based on the ARM CoreSight architecture and is configured to provide the maximum flexibility as allowed by the restrictions of the pinout and other available resources.

It provides register and memory accessibility from the external debugger interface, basic run/halt control plus 2 breakpoints and 2 watchpoints.

This device supports only one debug interface, Serial Wire Debug (SWD).

9.2 Debug port pin descriptions

The debug port pins default to their SWD functionality after power-on-reset (POR).

Table 9-1. Serial wire debug pin description

Pin Name	Type	Description
SWD_CLK	Input	Serial Wire Clock. This pin is the clock for debug logic when in the Serial Wire Debug mode. [1]
SWD_DIO	Input / Output	Serial Wire Debug Data input/output. The SWD_DIO pin is used by an external debug tool for communication and device control. This pin is pulled up internally.

1. The pad library of this device does not support on-chip pull down; the SWD_CLK pin supports only pullup controlled by PTAPE0, external pulldown resistor is required to fully support SWD protocol.

9.3 SWD status and control registers

Through the ARM Debug Access Port (DAP), the debugger has access to the status and control elements, implemented as registers on the DAP bus as shown in Figure 9-1. These registers provide additional control and status for low-power mode recovery and typical run-control scenarios. The status register bits also provide a means for the debugger to get updated status of the core without having to initiate a bus transaction across the crossbar switch, thus remaining less intrusive during a debug session.

A miscellaneous debug module (MDM) is implemented on this device, which contains the DAP control and status registers. It is important to note that these DAP control and status registers are not memory-mapped within the system memory map and are only accessible via the Debug Access Port using SWD. The MDM-AP is accessible as Debug Access Port 1 with the available registers shown in the table below.

Table 9-2. MDM-AP register summary

Address	Register	Description
0x0100_0000	Status	See MDM-AP status register
0x0100_0004	Control	See MDM-AP Control register
0x0100_00FC	IDR	Read-only identification register that always reads as 0x001C_0020

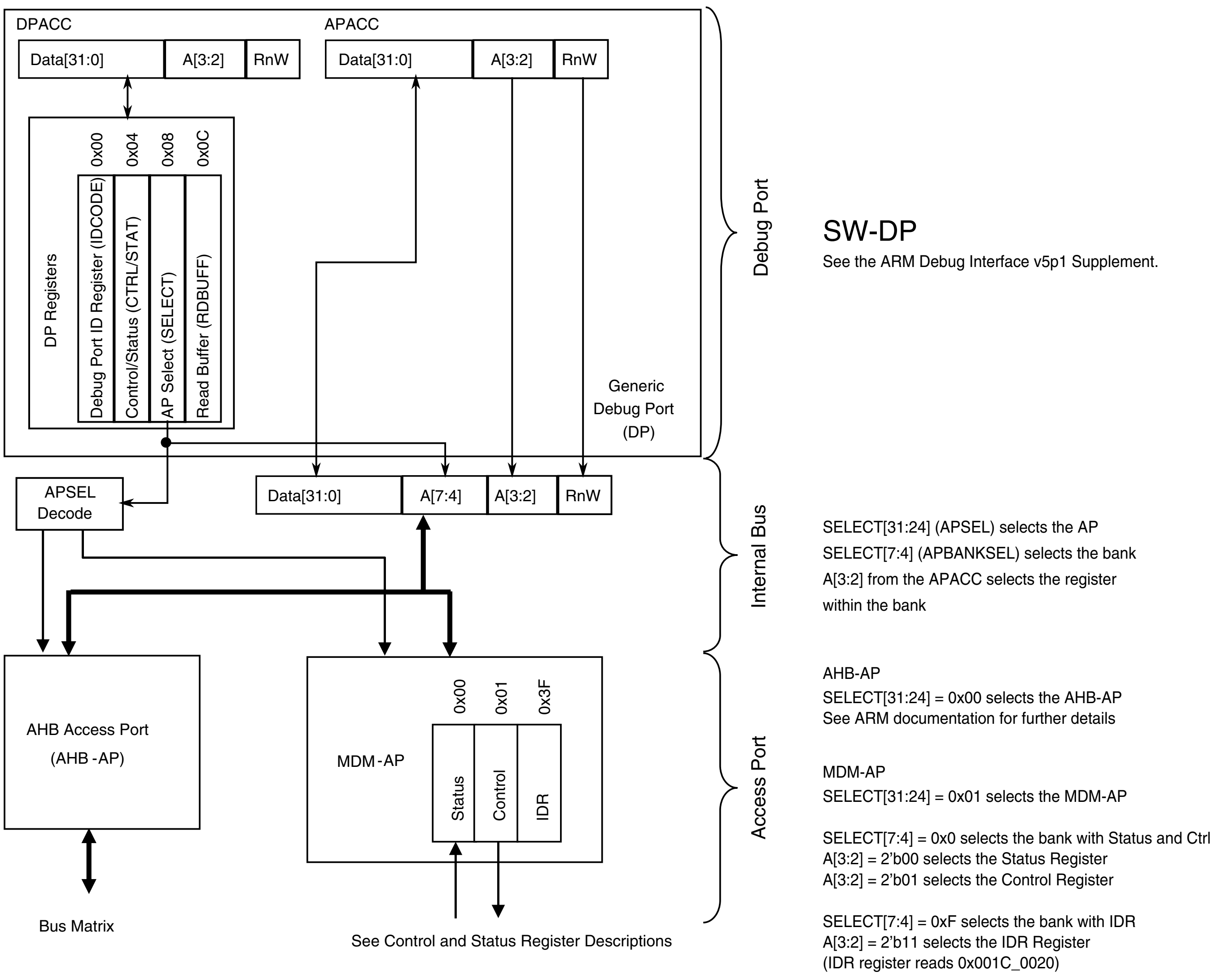

DPACC
Data[31:0]
A[3:2]
RnW
APACC
Data[31:0]
A[3:2]
RnW
DP Registers
0x00
0x04
0x08
0x0C
Debug Port ID Register (IDCODE)
Control/Status (CTRL/STAT)
AP Select (SELECT)
Read Buffer (RDBUFF)
Generic Debug Port (DP)
Debug Port
SW-DP
See the ARM Debug Interface v5p1 Supplement.
APSEL Decode
Data[31:0]
A[7:4]
A[3:2]
RnW
Internal Bus
SELECT[31:24] (APSEL) selects the AP
SELECT[7:4] (APBANKSEL) selects the bank
A[3:2] from the APACC selects the register within the bank
AHB Access Port (AHB -AP)
Bus Matrix
MDM -AP
0x00
0x01
0x3F
Status
Control
IDR
See Control and Status Register Descriptions
Access Port
AHB-AP
SELECT[31:24] = 0x00 selects the AHB-AP
See ARM documentation for further details
MDM-AP
SELECT[31:24] = 0x01 selects the MDM-AP
SELECT[7:4] = 0x0 selects the bank with Status and Ctrl
A[3:2] = 2'b00 selects the Status Register
A[3:2] = 2'b01 selects the Control Register
SELECT[7:4] = 0xF selects the bank with IDR
A[3:2] = 2'b11 selects the IDR Register
(IDR register reads 0x001C_0020)

Figure 9-1. MDM AP addressing

9.3.1 MDM-AP status register

Table 9-3. MDM-AP status register assignments

Bit	Name	Description
0	Flash Mass Erase Acknowledge	The Flash Mass Erase Acknowledge field is cleared after POR reset. The field is also cleared at launch of a mass erase command due to write of Flash Mass Erase in Progress field in MDM AP Control Register. The Flash Mass Erase Acknowledge is set after Flash control logic has started the mass erase operation.
1	Flash Ready	Indicates that flash memory has been initialized and debugger can be configured even if system is continuing to be held in reset via the debugger. 0 Flash is under initialization.

Table continues on the next page...

Table 9-3. MDM-AP status register assignments (continued)

Bit	Name	Description
		1 Flash is ready.
2	System Security	Indicates the security state. When secure, the debugger does not have access to the system bus or any memory mapped peripherals. This field indicates when the part is locked and no system bus access is possible. **NOTE:** This bit is not valid until Flash Ready bit set. 0 Device is unsecured. 1 Device is secured.
3	System Reset	Indicates the system reset state. 0 System is in reset. 1 System is not in reset.
4	Reserved	
5 – 15	Reserved for future use	Always read 0.
16	Core Halted	Indicates the core has entered Debug Halt mode 0 Core is not halted. 1 Core is halted.
17	Core SLEEPDEEP	SLEEPDEEP=1 indicates the core has entered Stop mode.
18	Core SLEEPING	SLEEPING=1 indicates the core has entered Wait mode.
19 – 31	Reserved for future use	Always reads 0.

9.3.2 MDM-AP Control register

Table 9-4. MDM-AP Control register assignments

Bit	Name	Secure[1]	Description
0	Flash Mass Erase in Progress	Y	Set to cause mass erase. Cleared by hardware after mass erase operation completes.
1	Debug Disable	N	Set to disable debug. Clear to allow debug operation. When set, it overrides the C_DEBUGEN field within the DHCSR[2] and forces to disable Debug logic.
2	Debug Request	N	Set to force the core to halt. If the core is in Stop or Wait mode, this field can be used to wake the core and transition to a halted state.
3	System Reset Request	Y	Set to force a system reset. The system remains held in reset until this field is cleared. When this bit is set, $\overline{\text{RESET}}$ pin does not reflect the status of system reset and does not keep low.
4	Core Hold	N	Configuration field to control core operation at the end of system reset sequencing. 0 Normal operation—release the core from reset along with the rest of the system at the end of system reset sequencing.

Table continues on the next page...

Table 9-4. MDM-AP Control register assignments (continued)

Bit	Name	Secure[1]	Description
			1 Suspend operation—hold the core in reset at the end of reset sequencing. Once the system enters this suspended state, clearing this control bit immediately releases the core from reset and CPU operation begins.
5– 31	Reserved for future use	N	

1. Command available in secure mode
2. DHCSR: refer to the Debug Halting Control and Status Register in the ARMv6-M Architecture Reference Mannual.

9.4 Debug resets

The debug system receives the following sources of reset:

- System POR reset

Conversely, the debug system is capable of generating system reset using the following mechanism:

- A system reset in the DAP control register which allows the debugger to hold the system in reset.
- Writing 1 to the SYSRESETREQ field in the NVIC Application Interrupt and Reset Control register
- A system reset in the DAP control register which allows the debugger to hold the core in reset.

9.5 Debug in low-power modes

In low-power modes in which the debug modules are kept static or powered off, the debugger cannot gather any debug data for the duration of the low-power mode.

- If the debugger is held static, the debug port returns to full functionality as soon as the low-power mode exits and the system returns to a state with active debug.
- If the debugger logic is powered off, the debugger is reset on recovery and must be reconfigured once the low-power mode is exited.

The active debug will prevent the chip from entering low-power mode. In case the chip is already in low-power mode, a debug request from MDM-AP control register will wake the chip from low-power mode.

9.6 Debug and security

When flash security is enabled, the debug port capabilities are limited in order to prevent exploitation of secure data. In the secure state, the debugger still has access to the status register and can determine the current security state of the device. In the case of a secure device, the debugger has the capability of performing only a mass erase operation.

Chapter 10
Signal Multiplexing and Signal Descriptions

10.1 Introduction

To optimize functionality in small packages, pins have several functions available via signal multiplexing. This chapter illustrates which of this device's signals are multiplexed on which external pin.

The Pin Selection Register (SIM_PINSEL) controls which signal is present on the external pin. Refer to that register to find the detailed control operation of a specific multiplexed pin.

10.2 Pinout

10.2.1 Signal Multiplexing and Pin Assignments

The following table shows the signals available on each pin and the locations of these pins on the devices supported by this document.

NOTE

- PTB5, PTC1, and PTC5 pins support high-current drive output, refer to the PORT_HDRVE register in Port Control chapter for details.
- VDD and VREFH are internally connected. Only one pin (VDD or VREFH) is available on chip.
- VSS and VREFL are internally connected. Only one pin (VSS or VREFL) is available on chip.
- PTA2 and PTA3 are true open-drain pins when operated as output

24 QFN	20 SOIC	16 TSSOP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
1	—	—	PTC5	DISABLED	PTC5	KBI1_P1	FTM2_CH3	BUSOUT				
2	—	—	PTC4	DISABLED	PTC4	KBI1_P0	FTM2_CH2		PWT_IN0			
3	3	3	VDD	VDD							VDD	
3	3	3	VREFH	VDDA/ VREFH						VDDA	VREFH	
4	4	4	VREFL	VREFL							VREFL	
4	4	4	VSS	VSS/ VSSA						VSSA	VSS	
5	5	5	PTB7	EXTAL	PTB7		I2C0_SCL				EXTAL	
6	6	6	PTB6	XTAL	PTB6		I2C0_SDA				XTAL	
7	7	7	PTB5	ACMP1_OUT	PTB5	KBI1_P7	FTM2_CH5	SPI0_PCS	ACMP1_OUT			
8	8	8	PTB4	NMI_b	PTB4	KBI1_P6	FTM2_CH4	SPI0_MISO	ACMP1_IN2	NMI_b		
9	9	—	PTC3	ADC0_SE11	PTC3	KBI1_P5	FTM2_CH3				ADC0_SE11	
10	10	—	PTC2	ADC0_SE10	PTC2	KBI1_P4	FTM2_CH2				ADC0_SE10	
11	11	—	PTC1	ADC0_SE9	PTC1	KBI1_P3	FTM2_CH1				ADC0_SE9	
12	12	—	PTC0	ADC0_SE8	PTC0	KBI1_P2	FTM2_CH0				ADC0_SE8	
13	13	9	PTB3	ADC0_SE7	PTB3	KBI0_P7	SPI0_MOSI	FTM0_CH1			ADC0_SE7	
14	14	10	PTB2	ADC0_SE6	PTB2	KBI0_P6	SPI0_SCK	FTM0_CH0	ACMP0_IN0		ADC0_SE6	
15	15	11	PTB1	ADC0_SE5	PTB1	KBI0_P5	UART0_TX	SPI0_MISO	TCLK2		ADC0_SE5	
16	16	12	PTB0	ADC0_SE4	PTB0	KBI0_P4	UART0_RX	SPI0_PCS	PWT_IN1		ADC0_SE4	
17	—	—	PTA7	ADC0_SE3	PTA7		FTM2_FLT2	SPI0_MOSI	ACMP1_IN1		ADC0_SE3	
18	—	—	PTA6	ADC0_SE2	PTA6		FTM2_FLT1	SPI0_SCK	ACMP1_IN0		ADC0_SE2	
19	17	13	PTA3	DISABLED	PTA3	KBI0_P3	UART0_TX	I2C0_SCL				
20	18	14	PTA2	DISABLED	PTA2	KBI0_P2	UART0_RX	I2C0_SDA				
21	19	15	PTA1	ADC0_SE1	PTA1	KBI0_P1	FTM0_CH1		ACMP0_IN1		ADC0_SE1	
22	20	16	PTA0	SWD_CLK	PTA0	KBI0_P0	FTM0_CH0	RTCO	ACMP0_IN2	ADC0_SE0	SWD_CLK	
23	1	1	PTA5	RESET_b	PTA5	IRQ	TCLK1				RESET_b	
24	2	2	PTA4	SWD_DIO	PTA4				ACMP0_OUT		SWD_DIO	

10.2.2 Device pin assignment

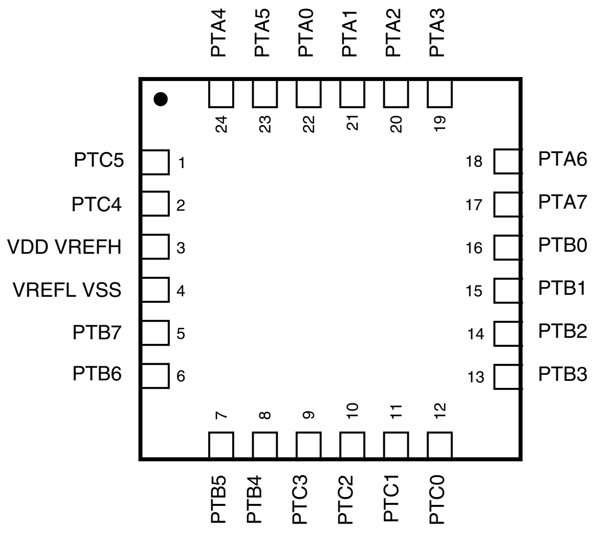


Figure 10-1. 24-pin QFN package

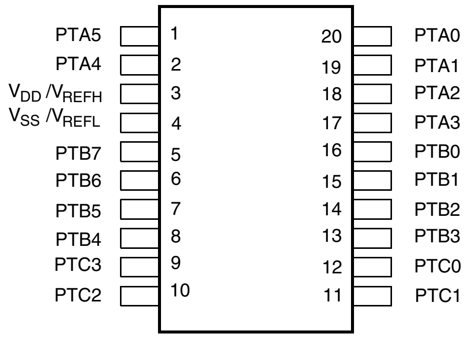


Figure 10-2. 20-pin SOIC package

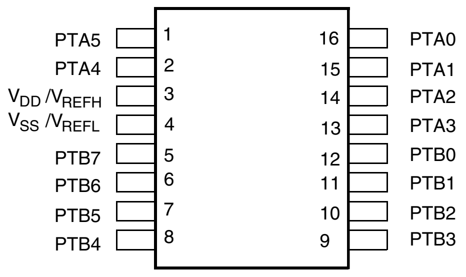


Figure 10-3. 16-pin TSSOP package

10.3 Module signal description tables

The following sections correlate the chip-level signal name with the signal name used in the module's chapter. They also briefly describe the signal function and direction.

10.3.1 Core modules

Table 10-1. SWD signal descriptions

Chip signal name	Module signal name	Description	I/O
SWD_DIO	SWD_DIO	Serial Wire Debug Data input/output. The SWD_DIO pin is used by an external debug tool for communication and device control. This pin is pulled up internally.	Input / Output
SWD_CLK	SWD_CLK	Serial Wire Clock. This pin is the clock for debug logic when in the Serial Wire Debug mode. [1]	Input

1. The pad library of this device does not support on-chip pull down; the SWD_CLK pin supports only pullup controlled by PTAPE0, external pulldown resistor is required to fully support SWD protocol.

10.3.2 System modules

Table 10-2. System signal descriptions

Chip signal name	Module signal name	Description	I/O
$\overline{\text{NMI}}$	—	Non-maskable interrupt **NOTE:** Driving the $\overline{\text{NMI}}$ signal low forces a non-maskable interrupt, if the $\overline{\text{NMI}}$ function is selected on the corresponding pin.	I

Table continues on the next page...

Table 10-2. System signal descriptions (continued)

Chip signal name	Module signal name	Description	I/O
$\overline{\text{RESET}}$	—	Reset bidirectional signal	I/O
VDD	—	MCU power	I
VSS	—	MCU ground	I

10.3.3 Clock modules

Table 10-3. OSC signal descriptions

Chip signal name	Module signal name	Description	I/O
EXTAL	EXTAL	External clock/oscillator input	Analog input
XTAL	XTAL	Oscillator output	Analog output

10.3.4 Analog

Table 10-4. ADC0 signal descriptions

Chip signal name	Module signal name	Description	I/O
ADC0_SEn	AD11-AD0	Analog channel inputs	I
VDD/VREFH	VDDA/VREFH	Analog power supply / voltage reference high	I
VSS/VREFL	VSSA/VREFL	Analog power ground / voltage reference low	I

Table 10-5. ACMP0 signal descriptions

Chip signal name	Module signal name	Description	I/O
ACMP0_INn	ACMP0_IN[2:0]	Analog voltage inputs	I
ACMP0_OUT	ACMP0_OUT	Comparator output	O

Table 10-6. ACMP1 signal descriptions

Chip signal name	Module signal name	Description	I/O
ACMP1_INn	ACMP1_IN[2:0]	Analog voltage inputs	I
ACMP1_OUT	ACMP1_OUT	Comparator output	O

10.3.5 Timer modules

Table 10-7. FTM0 signal descriptions

Chip signal name	Module signal name	Description	I/O
TCLKn	EXTCLK	FTM external clock	I
FTM0_CHn	CHn	FTM channel	I/O

Table 10-8. FTM2 signal descriptions

Chip signal name	Module signal name	Description	I/O
TCLKn	EXTCLK	FTM external clock	I
FTM2_CHn	CHn	FTM channel	I/O
FTM2_FLT1	FAULT1	Fault input (1)	I
FTM2_FLT2	FAULT2	Fault input (2)	I

Table 10-9. RTC signal descriptions

Chip signal name	Module signal name	Description	I/O
RTCO	RTCO	RTC clock output	O

Table 10-10. PWT signal descriptions

Chip signal name	Module signal name	Description	I/O
TCLKn	ALTCLK	PWT alternate external clock	O
PWT_IN0	PWTIN[0]	PWT input channel0	I
PWT_IN1	PWTIN[1]	PWT input channel1	I

10.3.6 Communication Interfaces

Table 10-11. SPI0 signal descriptions

Chip signal name	Module signal name	Description	I/O
SPI0_MISO	MISO	Master Data In, Slave Data Out	I/O
SPI0_MOSI	MOSI	Master Data Out, Slave Data In	I/O

Table continues on the next page...

Table 10-11. SPI0 signal descriptions (continued)

Chip signal name	Module signal name	Description	I/O
SPI0_SCK	SPSCK	SPI Serial Clock	I/O
SPI0_PCS	$\overline{SS}$	Slave Select	I/O

Table 10-12. I^{2}C0 signal descriptions

Chip signal name	Module signal name	Description	I/O
I2C0_SCL	SCL	Bidirectional serial clock line of the I^{2}C system.	I/O
I2C0_SDA	SDA	Bidirectional serial data line of the I^{2}C system.	I/O

Table 10-13. UART0 signal descriptions

Chip signal name	Module signal name	Description	I/O
UART0_TX	TxD	Transmit data	I/O
UART0_RX	RxD	Receive data	I

10.3.7 Human-machine interfaces (HMI)

Table 10-14. GPIO signal descriptions

Chip signal name	Module signal name	Description	I/O
PTA[7:0][1]	PORTA7-PORTA0	General-purpose input/output	I/O
PTB[7:0][1]	PORTA15-PORTA8	General-purpose input/output	I/O
PTC[5:0][1]	PORTA21-PORTA16	General-purpose input/output	I/O

1. The available GPIO pins depend on the specific package. See the signal multiplexing section for which exact GPIO signals are available.

Table 10-15. KBI0 signal descriptions

Chip signal name	Module signal name	Description	I/O
KBI0_Pn	KBI0Pn	Keyboard interrupt pins, n can be 0 ~ 7	I/O

Table 10-16. KBI1 signal descriptions

Chip signal name	Module signal name	Description	I/O
KBI1_Pn	KBI1Pn	Keyboard interrupt pins, n can be 0 ~ 7	I/O

Table 10-17. IRQ signal descriptions

Chip signal name	Module signal name	Description	I/O
IRQ	IRQ	IRQ input	I

Chapter 11
Port Control (PORT)

11.1 Introduction

This device has three sets of I/O ports, which include up to 22 general-purpose I/O pins.

Not all pins are available on all devices.

Many of the I/O pins are shared with on-chip peripheral functions. The peripheral modules have priority over the I/O, so when a peripheral is enabled, the associated I/O functions are disabled.

After reset, the shared peripheral functions are disabled so that the pins are controlled by the parallel I/O except PTA4, PTA0, PTB4 and PTA5 that are default to SWD_DIO, SWD_CLK, $\overline{\text{NMI}}$ and $\overline{\text{RESET}}$ function. All of the parallel I/O are configured as high-impedance (Hi-Z). The pin control functions for each pin are configured as follows:

- input disabled (GPIOx_PIDR[PID] = 1),
- output disabled (GPIOx_PDDR[PDD] = 0), and
- internal pullups disabled (PORT_PUE(L)[PTxPEn] = 0).

Additionally, the parallel I/O that support high drive capability are disabled (HDRVE = 0x00) after reset.

The following three figures show the structures of each I/O pin.

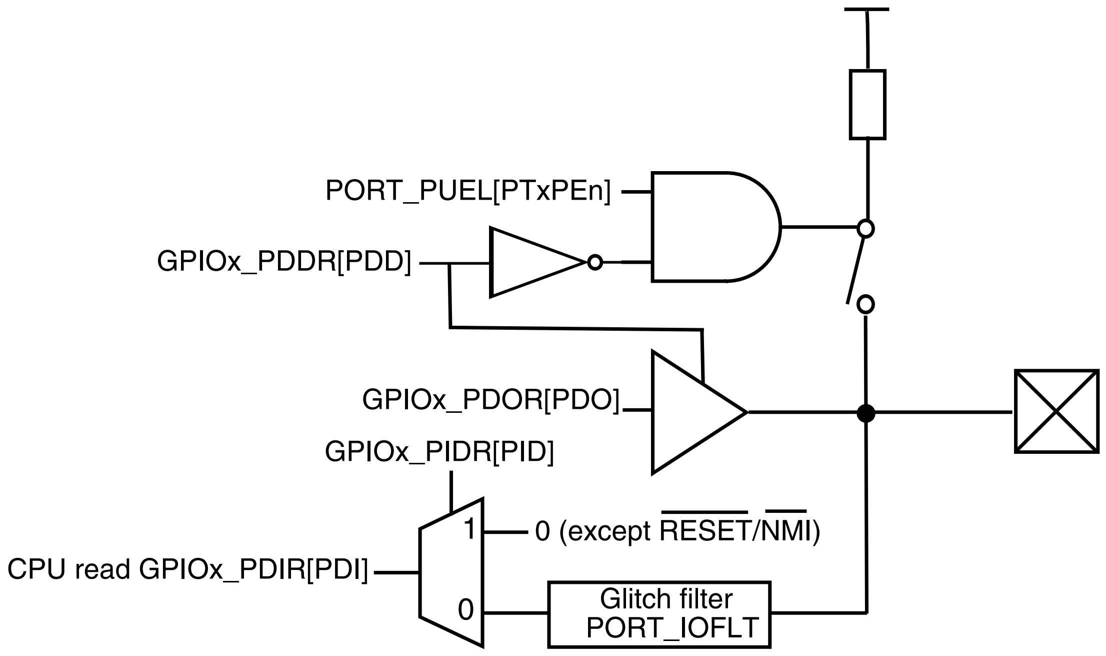


Figure 11-1. Normal I/O structure

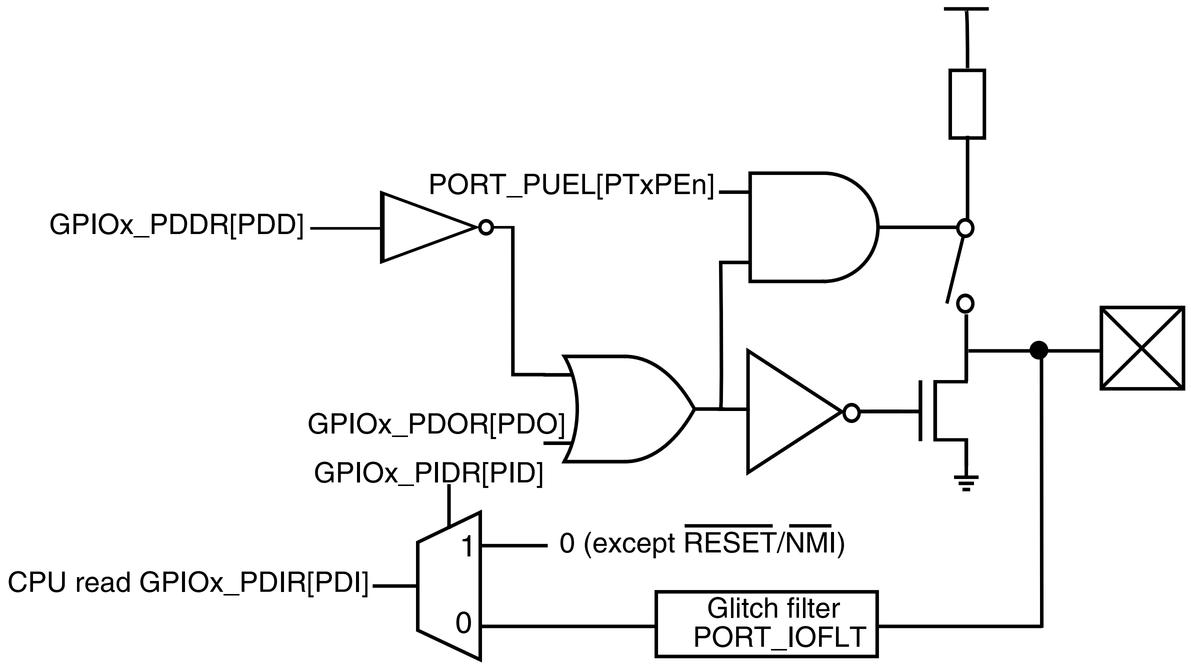


Figure 11-2. SDA(PTA2)/SCL(PTA3) structure

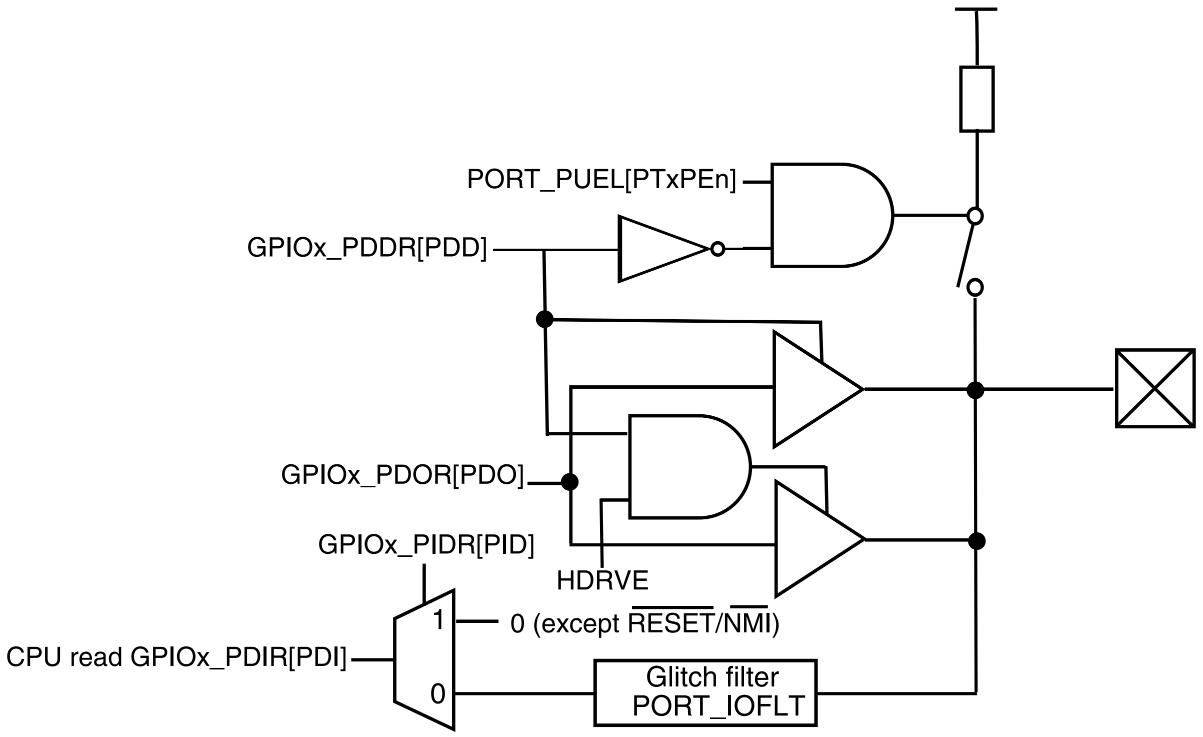


Figure 11-3. High drive I/O structure

11.2 Port data and data direction

Reading and writing of parallel I/O is accomplished through the port data registers (GPIOx_PDIR/PDOR). The direction, input or output, is controlled through the input disable register (GPIOx_PIDR) and data direction register (GPIOx_PDDR).

After reset, all parallel I/O default to the Hi-Z state. The corresponding bit in port data direction register (GPIOx_PDDR) or input disable register (GPIOx_PIDR) must be configured for output or input operation. Each port pin has an input disable bit and an output enable bit. When GPIOx_PIDR[PID] = 0, a read from GPIOx_PDIR returns the input value of the associated pin; when GPIOx_PIDR[PID] = 1, a read from GPIOx_PDIR[PDI] returns 0 except for $\overline{\text{RESET}}/\overline{\text{NMI}}$.

NOTE

The GPIOx_PDDR must be clear when the corresponding pin is used as input function to avoid contention. If set the corresponding GPIOx_PDDR and GPIOx_PIDR bits at same time, read from GPIOx_PDIR will always read the pin status.

When a peripheral module or system function is in control of a port pin, the data direction register bit still controls what is returned for reads of the port data register, even though the peripheral system has overriding control of the actual pin direction.

When a shared analog function is enabled for a pin, all digital pin functions are disabled. A read of the port data register returns a value of 0 for any bits that have shared analog functions enabled. In general, whenever a pin is shared with both an alternate digital function and an analog function, the analog function has priority such that if both of the digital and analog functions are enabled, the analog function controls the pin.

A write of valid data to a port data register must occur before setting the output enable bit of an associated port pin. This ensures that the pin will not be driven with an incorrect data value.

11.3 Internal pullup enable

An internal pullup device can be enabled for each port pin by setting the corresponding bit in one of the pullup enable registers (PORT_PUEL). The internal pullup device is disabled if the pin is configured as an output by the parallel I/O control logic, or by any shared peripheral function, regardless of the state of the corresponding pullup enable register bit. The internal pullup device is also disabled if the pin is controlled by an analog function.

NOTE

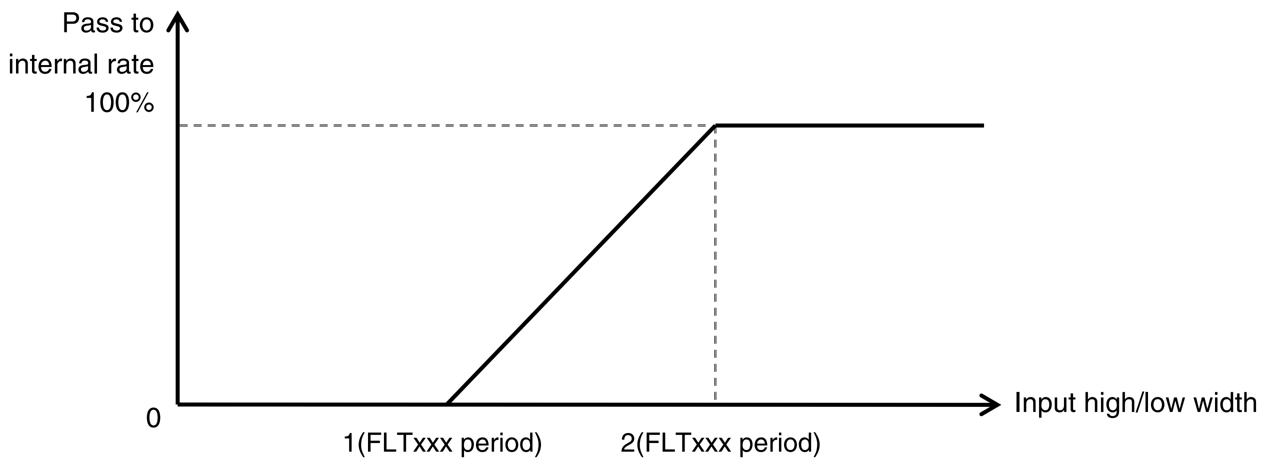
When configuring I2C0 to use "SDA(PTA2/PTB6) and SCL(PTA3/PTB7)" pins, and if an application uses internal pullups instead of external pullups, the internal pullups remain at present setting when the pins are configured as outputs, but they are automatically disabled to save power when the output values are low.

11.4 Input glitch filter setting

A filter is implemented for each port pin that is configured as a digital input. It can be used as a simple low-pass filter to filter any glitch that is introduced from the pins of GPIO, FTM0, PWT, IRQ,$\overline{\text{RESET}}$, $\overline{\text{NMI}}$ and KBI. The glitch width threshold can be adjusted easily by setting PORT_IOFLT[FLTDIVn] between 1~4096 BUSCLKs (or 1~128 LPOCLKs). This configurable glitch filter can take the place of an on board external analog filter, and greatly improve the EMC performance because any glitch will not be wrongly sampled or ignored.

Setting register PORT_IOFLT can configure the filters of the whole port or peripheral inputs. For example, setting PORT_IOFLT[FLTA] will affect all PTAn pins.

Glitches that are shorter than the selected clock period will be filtered out; Glitches that are more than twice the selected clock period will not be filtered out. It will pass to internal circuitry.



Note: FLTxxx is contents in register PORT_IOFLT.

Figure 11-4. Input glitch filter

11.5 High current drive

Output high sink/source current drive can be enabled by setting the corresponding bit in the HDRVE register for PTC5, PTC1, and PTB5. These pins can used as output and input; the pins output high sink/source current when they are operated as output.

- High-current drive function is disabled, if the pin is configured as an input by the parallel I/O control logic.
- When configured as any shared peripheral function, high-current drive function still works on these pins, but only when they are configured as outputs.

11.6 Pin behavior in Stop mode

In Stop mode, all I/O is maintained because internal logic circuitry stays powered up. Upon recovery, normal I/O function is available to the user.

11.7 Port data registers

PORT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_9000	Port Filter Register (PORT_IOFLT)	32	R/W	00C0_0000h	11.7.1/146
4004_9004	Port Pullup Enable Low Register (PORT_PUEL)	32	R/W	0000_0001h	11.7.2/149
4004_900C	Port High Drive Enable Register (PORT_HDRVE)	32	R/W	0000_0000h	11.7.3/152

11.7.1 Port Filter Register (PORT_IOFLT)

This register sets the filters for input pins. Configure the high/low level glitch width threshold. Glitches that are shorter than the selected clock period will be filtered out; glitches that are more than twice the selected clock period will not be filtered out and will pass to internal circuitry.

Address: 4004_9000h base + 0h offset = 4004_9000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FLTDIV3			FLTDIV2			FLTDIV1		FLTNMI		FLTKBI1		FLTKBI0		FLTRST	
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLTPWT		FLTFTM0		FLTIIC		0				FLTC		FLTB		FLTA	
W							▨	▨	▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PORT_IOFLT field descriptions

Field	Description
31–29 FLTDIV3	Filter Division Set 3 Port Filter Division Set 3 000 LPOCLK 001 LPOCLK/2 010 LPOCLK/4 011 LPOCLK/8 100 LPOCLK/16 101 LPOCLK/32 110 LPOCLK/64 111 LPOCLK/128
28–26 FLTDIV2	Filter Division Set 2

Table continues on the next page...

PORT_IOFLT field descriptions (continued)

Field	Description
	Port Filter Division Set 2 000 BUSCLK/32 001 BUSCLK/64 010 BUSCLK/128 011 BUSCLK/256 100 BUSCLK/512 101 BUSCLK/1024 110 BUSCLK/2048 111 BUSCLK/4096
25–24 FLTDIV1	Filter Division Set 1 Port Filter Division Set 1 00 BUSCLK/2 01 BUSCLK/4 10 BUSCLK/8 11 BUSCLK/16
23–22 FLTNMI	Filter Selection for Input from NMI 00 No filter. 01 Selects FLTDIV1, and will switch to FLTDIV3 in Stop mode automatically. 10 Selects FLTDIV2, and will switch to FLTDIV3 in Stop mode automatically. 11 FLTDIV3
21–20 FLTKBI1	Filter Selection for Input from KBI1 00 No filter 01 Selects FLTDIV1, and will switch to FLTDIV3 in Stop mode automatically. 10 Selects FLTDIV2, and will switch to FLTDIV3 in Stop mode automatically. 11 FLTDIV3
19–18 FLTKBI0	Filter selection for Input from KBI0 00 No filter. 01 Selects FLTDIV1, and will switch to FLTDIV3 in Stop mode automatically. 10 Selects FLTDIV2, and will switch to FLTDIV3 in Stop mode automatically. 11 FLTDIV3
17–16 FLTRST	Filter Selection for Input from RESET/IRQ 00 No filter. 01 Selects FLTDIV1, and will switch to FLTDIV3 in Stop mode automatically. 10 Selects FLTDIV2, and will switch to FLTDIV3 in Stop mode automatically. 11 FLTDIV3
15–14 FLTPWT	Filter Selection For Input from PWT_IN1/PWT_IN0 00 No filter 01 Select FLTDIV1 10 Select FLTDIV2 11 Select FLTDIV3

Table continues on the next page...

PORT_IOFLT field descriptions (continued)

Field	Description
13–12 FLTFTM0	Filter Selection For Input from FTM0CH0/FTM0CH1 00 No filter 01 Select FLTDIV1 10 Select FLTDIV2 11 Select FLTDIV3
11–10 FLTIIC	Filter Selection For Input from SCL/SDA 00 No filter 01 Select FLTDIV1 10 Select FLTDIV2 11 Select BUSCLK
9–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 FLTC	Filter Selection for Input from PTC 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
3–2 FLTB	Filter Selection for Input from PTB 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
FLTA	Filter Selection for Input from PTA 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3

11.7.2 Port Pullup Enable Low Register (PORT_PUEL)

Address: 4004_9000h base + 4h offset = 4004_9004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PORT_PUEL field descriptions

Field	Description
31–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PTCPE5	Pull Enable for Port C Bit 5 This control field determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port C bit 5. 1 Pullup is enabled for port C bit 5.
20 PTCPE4	Pull Enable for Port C Bit 4 This control field determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port C bit 4. 1 Pullup is enabled for port C bit 4.
19 PTCPE3	Pull Enable for Port C Bit 3 This control field determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port C bit 3. 1 Pullup is enabled for port C bit 3.
18 PTCPE2	Pull Enable for Port C Bit 2 This control field determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port C bit 2. 1 Pullup is enabled for port C bit 2.

Table continues on the next page...

PORT_PUEL field descriptions (continued)

Field	Description
17 PTCPE1	Pull Enable for Port C Bit 1 This control field determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port C bit 1. 1 Pullup is enabled for port C bit 1.
16 PTCPE0	Pull Enable for Port C Bit 0 This control field determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port C bit 0. 1 Pullup is enabled for port C bit 0.
15 PTBPE7	Pull Enable for Port B Bit 7 This control field determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port B bit 7. 1 Pullup is enabled for port B bit 7.
14 PTBPE6	Pull Enable for Port B Bit 6 This control field determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port B bit 6. 1 Pullup is enabled for port B bit 6.
13 PTBPE5	Pull Enable for Port B Bit 5 This control field determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port B bit 5. 1 Pullup is enabled for port B bit 5.
12 PTBPE4	Pull Enable for Port B Bit 4 This control field determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port B bit 4. 1 Pullup is enabled for port B bit 4.
11 PTBPE3	Pull Enable for Port B Bit 3 This control field determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port B bit 3. 1 Pullup is enabled for port B bit 3.
10 PTBPE2	Pull Enable for Port B Bit 2

Table continues on the next page...

PORT_PUEL field descriptions (continued)

Field	Description
	This control field determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port B bit 2. 1 Pullup is enabled for port B bit 2.
9 PTBPE1	Pull Enable for Port B Bit 1 This control field determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs or Hi-Z, tthis field has no effect. 0 Pullup is disabled for port B bit 1. 1 Pullup is enabled for port B bit 1.
8 PTBPE0	Pull Enable for Port B Bit 0 This control field determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port B bit 0. 1 Pullup is enabled for port B bit 0.
7 PTAPE7	Pull Enable for Port A Bit 7 This control field determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port A bit 7. 1 Pullup is enabled for port A bit 7.
6 PTAPE6	Pull Enable for Port A Bit 6 This control field determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port A bit 6. 1 Pullup is enabled for port A bit 6.
5 PTAPE5	Pull Enable for Port A Bit 5 This control field determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port A bit 5. 1 Pullup is enabled for port A bit 5.
4 PTAPE4	Pull Enable for Port A Bit 4 This control field determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port A bit 4. 1 Pullup is enabled for port A bit 4.
3 PTAPE3	Pull Enable for Port A Bit 3 This control field determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs or Hi-Z, this field has no effect.

Table continues on the next page...

PORT_PUEL field descriptions (continued)

Field	Description
	NOTE: When configuring to use this pin as output high for IIC, the internal pullup device remains active when PTAPE3 is set. It is automatically disabled to save power when output low. 0 Pullup is disabled for port A bit 3. 1 Pullup is enabled for port A bit 3.
2 PTAPE2	Pull Enable for Port A Bit 2 This control field determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs or Hi-Z, this field has no effect. **NOTE:** When configuring to use this pin as output high for IIC, the internal pullup device remains active when PTAPE2 is set. It is automatically disabled to save power when output low. 0 Pullup is disabled for port A bit 2. 1 Pullup is enabled for port A bit 2.
1 PTAPE1	Pull Enable for Port A Bit 1 This control field determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port A bit 1. 1 Pullup is enabled for port A bit 1.
0 PTAPE0	Pull Enable for Port A Bit 0 This control field determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs or Hi-Z, this field has no effect. 0 Pullup is disabled for port A bit 0. 1 Pullup is enabled for port A bit 0.

11.7.3 Port High Drive Enable Register (PORT_HDRVE)

Address: 4004_9000h base + Ch offset = 4004_900Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												PTC5	PTC1	PTB5	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PORT_HDRVE field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PORT_HDRVE field descriptions (continued)

Field	Description
3 PTC5	High Current Drive Capability of PTC5 This read/write field enables the high current drive capability of PTC5. 0 PTC5 is disabled to offer high current drive capability. 1 PTC5 is enabled to offer high current drive capability.
2 PTC1	High Current Drive Capability of PTC1 This read/write field enables the high current drive capability of PTC1 0 PTC1 is disabled to offer high current drive capability. 1 PTC1 is enabled to offer high current drive capability.
1 PTB5	High Current Drive Capability of PTB5 This read/write field enables the high current drive capability of PTB5 0 PTB5 is disabled to offer high current drive capability. 1 PTB5 is enabled to offer high current drive capability.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Chapter 12
System Integration Module (SIM)

12.1 Introduction

The system integration module (SIM) provides system control and chip configuration registers.

12.1.1 Features

The features of the SIM module are listed below.

- Reset status and device ID information
- System interconnection configuration and special pin enable
- Pin re-map control
- System clock gating control and clock divide

12.2 Memory map and register definition

The SIM module contains many fields for selecting the clock source and dividers for various module clocks.

SIM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_8000	System Reset Status and ID Register (SIM_SRSID)	32	R	See section	12.2.1/156
4004_8004	System Options Register (SIM_SOPT)	32	R/W	See section	12.2.2/159
4004_8008	Pin Selection Register (SIM_PINSEL)	32	R/W	0000_0000h	12.2.3/162
4004_800C	System Clock Gating Control Register (SIM_SCGC)	32	R/W	0000_3000h	12.2.4/164
4004_8010	Universally Unique Identifier Low Register (SIM_UUIDL)	32	R	Undefined	12.2.5/167

Table continues on the next page...

SIM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4004_8014	Universally Unique Identifier Middle Low Register (SIM_UUIDML)	32	R	Undefined	12.2.6/168
4004_8018	Universally Unique Identifier Middle High Register (SIM_UUIDMH)	32	R	Undefined	12.2.7/168
4004_801C	Clock Divider Register (SIM_CLKDIV)	32	R/W	See section	12.2.8/169

12.2.1 System Reset Status and ID Register (SIM_SRSID)

Address: 4004_8000h base + 0h offset = 4004_8000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FAMID				SUBFAMID				RevID				PINID			
W																
Reset	0	0	0	0	0	1	0	0	*	*	*	*	*	*	*	*
LVD	0	0	0	0	0	1	0	0	*	*	*	*	*	*	*	*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		SACKERR	0	MDMAP	SW	LOCKUP	0	POR	PIN	WDOG	0		LOC	LVD	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
LVD	0	0	0	0	0	0	0	0	u*	0	0	0	0	0	1	0

* Notes:
- RevID field: Decided by device revision number.
- PINID field: Decided by device pin number.
- u = Unaffected by reset.

SIM_SRSID field descriptions

Field	Description
31–28 FAMID	Kinetis family ID 0000 KE0x family. other Reserved.
27–24 SUBFAMID	Kinetis sub-family ID 0100 KEx4 sub-family other Reserved
23–20 RevID	Device Revision Number
19–16 PINID	Device Pin ID 0000 8-pin 0001 16-pin 0010 20-pin 0011 24-pin 0100 32-pin 0101 44-pin 0110 48-pin 0111 64-pin 1000 80-pin 1010 100-pin other Reserved
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SACKERR	Stop Mode Acknowledge Error Reset Indicates that after an attempt to enter Stop mode, a reset has been caused by the failure of one or more IICs to acknowledge within approximately one second to enter stop mode. 0 Reset is not caused by peripheral failure to acknowledge attempt to enter Stop mode. 1 Reset is caused by peripheral failure to acknowledge attempt to enter Stop mode.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MDMAP	MDM-AP System Reset Request Indicates a reset has been caused by the host debugger system setting of the System Reset Request field in the MDM-AP Control Register. 0 Reset is not caused by host debugger system setting of the System Reset Request bit. 1 Reset is caused by host debugger system setting of the System Reset Request bit.
10 SW	Software Indicates a reset has been caused by software setting of SYSRESETREQ bit in Application Interrupt and Reset Control Register in the ARM core. 0 Reset is not caused by software setting of SYSRESETREQ bit. 1 Reset caused by software setting of SYSRESETREQ bit
9 LOCKUP	Core Lockup

Table continues on the next page...

SIM_SRSID field descriptions (continued)

Field	Description
	Indicates a reset has been caused by the ARM core indication of a LOCKUP event. 0 Reset is not caused by core LOCKUP event. 1 Reset is caused by core LOCKUP event.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 POR	Power-On Reset Causes reset by the power-on detection logic. When the internal supply voltage is ramping up, the low-voltage reset (LVR) status field is also set at that time, to indicate that the reset has occurred while the internal supply was below the LVR threshold. **NOTE:** This bit POR to 1, LVR to uncertain value and reset to 0 at any other conditions. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	External Reset Pin Causes reset by an active low-level on the external reset pin. 0 Reset is not caused by external reset pin. 1 Reset came from external reset pin.
5 WDOG	Watchdog (WDOG) Causes reset by the WDOG timer timing out. This reset source may be blocked by WDOG_CS1[EN] = 0. 0 Reset is not caused by WDOG timeout. 1 Reset is caused by WDOG timeout.
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 LOC	Internal Clock Source Module Reset Causes reset by an ICS module reset. 0 Reset is not caused by the ICS module. 1 Reset is caused by the ICS module.
1 LVD	Low Voltage Detect If PMC_SPMSC1[LVDRE] is set in Run mode or both PMC_SPMSC1[LVDRE] and PMC_SPMSC1[LVDSE] are set in Stop mode, and the supply drops below the LVD trip voltage, an LVD reset will occur. This field is also set by POR. **NOTE:** This field is reset to 1 on POR and LVR, and reset to 0 on other reset. 0 Reset is not caused by LVD trip or POR. 1 Reset is caused by LVD trip or POR.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.2 System Options Register (SIM_SOPT)

NOTE

RSTPE and NMIE are write-once only on each reset.

Address: 4004_8000h base + 4h offset = 4004_8004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DELAY								DLYACT	ADHWT			CLKOE	BUSREF		
W									▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
POR/ LVD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXDME	0	0	RXDCE	0		RXDFE		FTMIC		ACTRG	0	SWDE	RSTPE	NMIE	0
W		FTMSYNC	▨		▨							▨				▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	u*	u*	0
POR/ LVD	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

* Notes:
- u = Unaffected by reset.

SIM_SOPT field descriptions

Field	Description
31–24 DELAY	FTM2 Trigger Delay Specifies the delay from FTM2 initial or match trigger to ADC hardware trigger when 1 is written to ADHWT. The 8-bit modulo value allows the delay from 0 to 255 upon the BUSREF clock settings. This is a

Table continues on the next page...

SIM_SOPT field descriptions (continued)

Field	Description
	one-shot counter that starts ticking when the trigger arrives and stops ticking when the counter value reaches the modulo value that is defined.
23 DLYACT	FTM2 Trigger Delay Active This read-only field specifies the status if the FTM2 initial or match delay is active. This field is set when an FTM2 trigger arrives and the delay counter is ticking. Otherwise, this field will be clear. 0 The delay is inactive. 1 The delay is active.
22–20 ADHWT	ADC Hardware Trigger Source Selects the ADC hardware trigger source. All trigger sources start ADC conversion on rising-edge. 000 RTC overflow as the ADC hardware trigger 001 FTM0 init trigger as the ADC hardware trigger 010 FTM2 init trigger with 8-bit programmable counter delay 011 FTM2 match trigger with 8-bit programmable counter delay 100 PIT channel0 overflow as the ADC hardware trigger 101 PIT channel1 overflow as the ADC hardware trigger 110 ACMP0 out as the ADC hardware trigger. 111 ACMP1 out as the ADC hardware trigger
19 CLKOE	Bus Clock Output Enable Enables bus clock output on 0 Bus clock output is disabled on PTC5. 1 Bus clock output is enabled on PTC5.
18–16 BUSREF	BUS Clock Output select Enables bus clock output via an optional prescaler. 000 Bus 001 Bus divided by 2 010 Bus divided by 4 011 Bus divided by 8 100 Bus divided by 16 101 Bus divided by 32 110 Bus divided by 64 111 Bus divided by 128
15 TXDME	UART0_TX Modulation Select Enables the UART0_TX output modulated by FTM0 channel 0. 0 UART0_TX output is connected to pinout directly. 1 UART0_TX output is modulated by FTM0 channel 0 before mapped to pinout.
14 FTMSYNC	FTM2 Synchronization Select Generates a PWM synchronization trigger to the FTM2 module if 1 is written to this field. 0 No synchronization triggered. 1 Generates a PWM synchronization trigger to the FTM2 modules.

Table continues on the next page...

SIM_SOPT field descriptions (continued)

Field	Description
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 RXDCE	UART0_RX Capture Select Enables the UART0_RX to be captured by FTM0 channel 1. 0 UART0_RX input signal is connected to the UART0 module only. 1 UART0_RX input signal is connected to the UART0 module and FTM0 channel 1.
11–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–8 RXDFE	UART0 RxD Filter Select Enables the UART0 RxD input to be filtered by ACMP. When this function is enabled, any signal tagged with ACMP inputs can be regarded UART0. 00 RXD0 input signal is connected to UART0 module directly. 01 RXD0 input signal is filtered by ACMP0, then injected to UART0. 10 RXD0 input signal is filtered by ACMP1, then injected to UART0. 11 Reserved.
7–6 FTMIC	FTM0CH0 Input Capture Source Selects the sources for FTM0CH0 as capture input. 00 FTM0_CH0 pin 01 ACMP0 OUT 10 ACMP1 OUT 11 RTC overflow
5 ACTRG	ACMP Trigger FTM2 selection Selects the two ACMP outputs as the trigger0 input of FTM2 0 ACMP0 out 1 ACMP1 out
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 SWDE	Single Wire Debug Port Pin Enable Enables the PTA4/ACMP0_OUT/SWD_DIO pin to function as SWD_DIO, and PTA0/KBI0_P0/FTM0_CH0/RTCO/ACMP0_IN2/ADC0_SE0/SWD_CLK pin function as SWD_CLK. When clear, the two pins function as PTA4 and PTA0. This pin defaults to the SWD_DIO and SWD_CLK function following any MCU reset. 0 PTA4/ACMP0_OUT/SWD_DIO as PTA4 or ACMP0_OUT function, PTA0/KBI0_P0/FTM0_CH0/RTCO/ACMP0_IN2/ADC0_SE0/SWD_CLK as PTA0, KBI0_P0, FTM0_CH0, RTCO, ACMP0_IN2 or ADC0_SE0 function. 1 PTA4/ACMP0_OUT/SWD_DIO as SWD_DIO function, PTA0/KBI0_P0/FTM0_CH0/RTCO/ACMP0_IN2/ADC0_SE0/SWD_CLK as SWD_CLK function.
2 RSTPE	RESET Pin Enable This write-once field can be written after any reset. When RSTPE is set, the PTA5/IRQ/TCLK1/$\overline{\text{RESET}}$ pin functions as $\overline{\text{RESET}}$. When clear, the pin functions as one of its alternative functions. This pin defaults to

Table continues on the next page...

SIM_SOPT field descriptions (continued)

Field	Description
	$\overline{RESET}$ following an MCU POR. Other resets will not affect this field. When RSTPE is set, an internal pullup device on $\overline{RESET}$ is enabled. 0 PTA5/IRQ/TCLK1/$\overline{RESET}$ pin functions as PTA5, IRQ, or TCLK1. 1 PTA5/IRQ/TCLK1/$\overline{RESET}$ pin functions as $\overline{RESET}$.
1 NMIE	NMI Pin Enable This write-once field can be written after any reset. When NMIE is set, the PTB4/KBI1_P6/FTM2_CH4/SPI0_MISO/ACMP1_IN2/$\overline{NMI}$ pin functions as $\overline{NMI}$. When clear, the pin functions as one of its alternative functions. This pin defaults to $\overline{NMI}$ following an MCU POR. Other resets will not affect this bit. When NMIE is set, an internal pullup device on $\overline{NMI}$ is enabled. 0 PTB4/KBI1_P6/FTM2_CH4/SPI0_MISO/ACMP1_IN2/$\overline{NMI}$ pin functions as PTB4, KBI1_P6, FTM2_CH4, SPI0_MISO or ACMP1_IN2. 1 PTB4/KBI1_P6/FTM2_CH4/SPI0_MISO/ACMP1_IN2/$\overline{NMI}$ pin functions as $\overline{NMI}$.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.3 Pin Selection Register (SIM_PINSEL)

Address: 4004_8000h base + 8h offset = 4004_8008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PWTCLKPS	FTM2CLKPS	0	FTM0CLKPS	0											
W			▨		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FTM2PS3	FTM2PS2	0				FTM0PS1	FTM0PS0	UART0PS	SPI0PS	I2C0PS	0				
W			▨	▨	▨	▨						▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIM_PINSEL field descriptions

Field	Description
31 PWTCLKPS	PWT TCLK Pin Select Selects the TCLK pinout. 0 Selects TCLK1 for PWT module. 1 Selects TCLK2 for PWT module.

Table continues on the next page...

SIM_PINSEL field descriptions (continued)

Field	Description
30 FTM2CLKPS	FTM2 TCLK Pin Select Selects the TCLK pinout. 0 Selects TCLK1 for FTM2 module. 1 Selects TCLK2 for FTM2 module.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 FTM0CLKPS	FTM0 TCLK Pin Select Selects the TCLK pinout. 0 Selects TCLK1 for FTM0 module. 1 Selects TCLK2 for FTM0 module.
27–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 FTM2PS3	FTM2_CH3 Port Pin Select Selects the FTM2_CH3 channel pinout. 0 FTM2_CH3 channels are mapped on PTC3. 1 FTM2_CH3 channels are mapped on PTC5.
14 FTM2PS2	FTM2_CH2 Port Pin Select Selects the FTM2_CH2 channel pinout. 0 FTM2_CH2 channels are mapped on PTC2. 1 FTM2_CH2 channels are mapped on PTC4.
13–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 FTM0PS1	FTM0_CH1 Port Pin Select Selects the FTM0_CH1 channel pinout. 0 FTM0_CH1 channels are mapped on PTA1. 1 FTM0_CH1 channels are mapped on PTB3.
8 FTM0PS0	FTM0_CH0 Port Pin Select Selects the FTM0_CH0 channel pinout. 0 FTM0_CH0 channels are mapped on PTA0. 1 FTM0_CH0 channels are mapped on PTB2.
7 UART0PS	UART0 Pin Select Selects the UART0 pinouts. 0 UART0_RX and UART0_TX are mapped on PTB0 and PTB1. 1 UART0_RX and UART0_TX are mapped on PTA2 and PTA3.
6 SPI0PS	SPI0 Pin Select Selects the SPI0 Pinouts.

Table continues on the next page...

SIM_PINSEL field descriptions (continued)

Field	Description
	0 SPI0_SCK, SPI0_MOSI, SPI0_MISO, and SPI0_PCS are mapped on PTB2, PTB3, PTB4, and PTB5. 1 SPI0_SCK, SPI0_MOSI, SPI0_MISO, and SPI0_PCS are mapped on PTA6, PTA7, PTB1, and PTB0.
5 I2C0PS	I2C0 Port Pin Select Selects the I2C0 port pins. 0 I2C0_SCL and I2C0_SDA are mapped on PTA3 and PTA2, respectively. 1 I2C0_SCL and I2C0_SDA are mapped on PTB7 and PTB6, respectively.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.2.4 System Clock Gating Control Register (SIM_SCGC)

Address: 4004_8000h base + Ch offset = 4004_800Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ACMP1	ACMP0	ADC	0	IRQ	0	KBI1	KBI0	0	0		UART0	0	SPI0	I2C	0
W				▨		▨			▨	▨			▨			▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		SWD	FLASH	0	CRC	0		FTM2	0	FTM0	PWT	0		PIT	RTC
W	▨				▨		▨			▨			▨			
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

SIM_SCGC field descriptions

Field	Description
31 ACMP1	ACMP1 Clock Gate Control Controls the clock gate to the ACMP1 module. 0 Bus clock to the ACMP1 module is disabled. 1 Bus clock to the ACMP1 module is enabled.
30 ACMP0	ACMP0 Clock Gate Control Controls the clock gate to the ACMP0 module. 0 Bus clock to the ACMP0 module is disabled. 1 Bus clock to the ACMP0 module is enabled.
29 ADC	ADC Clock Gate Control Controls the clock gate to the ADC module.

Table continues on the next page...

SIM_SCGC field descriptions (continued)

Field	Description
	0 Bus clock to the ADC module is disabled. 1 Bus clock to the ADC module is enabled.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 IRQ	IRQ Clock Gate Control Controls the clock gate to the IRQ module. 0 Bus clock to the IRQ module is disabled. 1 Bus clock to the IRQ module is enabled.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 KBI1	KBI1 Clock Gate Control Controls the clock gate to the KBI1 module. 0 Bus clock to the KBI1 module is disabled. 1 Bus clock to the KBI1 module is enabled.
24 KBI0	KBI0 Clock Gate Control Controls the clock gate to the KBI0 module. 0 Bus clock to the KBI0 module is disabled. 1 Bus clock to the KBI0 module is enabled.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 UART0	UART0 Clock Gate Control Controls the clock gate to the UART0 module. 0 Bus clock to the UART0 module is disabled. 1 Bus clock to the UART0 module is enabled.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SPI0	SPI0 Clock Gate Control Controls the clock gate to the SPI0 module. 0 Bus clock to the SPI0 module is disabled. 1 Bus clock to the SPI0 module is enabled.
17 I2C	I2C Clock Gate Control Controls the clock gate to the I2C module. 0 Bus clock to the IIC module is disabled. 1 Bus clock to the IIC module is enabled.
16–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIM_SCGC field descriptions (continued)

Field	Description
13 SWD	SWD (single wire debugger) Clock Gate Control Controls the clock gate to the SWD module. 0 Bus clock to the SWD module is disabled. 1 Bus clock to the SWD module is enabled.
12 FLASH	Flash Clock Gate Control Controls the clock gate to the flash module. 0 Bus clock to the flash module is disabled. 1 Bus clock to the flash module is enabled.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 CRC	CRC Clock Gate Control Controls the clock gate to the CRC module. 0 Bus clock to the CRC module is disabled. 1 Bus clock to the CRC module is enabled.
9–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 FTM2	FTM2 Clock Gate Control Controls the clock gate to the FTM2 module. 0 Bus clock to the FTM2 module is disabled. 1 Bus clock to the FTM2 module is enabled.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 FTM0	FTM0 Clock Gate Control Controls the clock gate to the FTM0 module. 0 Bus clock to the FTM0 module is disabled. 1 Bus clock to the FTM0 module is enabled.
4 PWT	PWT Clock Gate Control Controls the clock gate to the PWT module. 0 Timer clock to the PWT module is disabled. 1 Timer clock to the PWT module is enabled.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 PIT	PIT Clock Gate Control Controls the clock gate to the PIT module. 0 Bus clock to the PIT module is disabled. 1 Bus clock to the PIT module is enabled.

Table continues on the next page...

SIM_SCGC field descriptions (continued)

Field	Description
0 RTC	RTC Clock Gate Control Controls the clock gate to the RTC module. 0 Bus clock to the RTC module is disabled. 1 Bus clock to the RTC module is enabled.

12.2.5 Universally Unique Identifier Low Register (SIM_UUIDL)

The read-only SIM_UUIDL register contains a series of number to identify the unique device in the family.

Address: 4004_8000h base + 10h offset = 4004_8010h

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

* Notes:

- x = Undefined at reset.

SIM_UUIDL field descriptions

Field	Description
ID[31:0]	Universally Unique Identifier

12.2.6 Universally Unique Identifier Middle Low Register (SIM_UUIDML)

The read-only SIM_UUIDML register contains a series of number to identify the unique device in the family.

Address: 4004_8000h base + 14h offset = 4004_8014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ID[63:32]																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

SIM_UUIDML field descriptions

Field	Description
ID[63:32]	Universally Unique Identifier

12.2.7 Universally Unique Identifier Middle High Register (SIM_UUIDMH)

The read-only SIM_UUIDMH register contains a series of number to identify the unique device in the family.

Address: 4004_8000h base + 18h offset = 4004_8018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																ID[80:64]															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

SIM_UUIDMH field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ID[80:64]	Universally Unique Identifier

12.2.8 Clock Divider Register (SIM_CLKDIV)

This register sets the divide value for the clock.

NOTE

Carefully confiugre the OUTDIV1 and OUTDIV2 to avoid bus clock frequency high than 24 MHz.

While configuring the OUTDIV1, OUTDIV2 and OUTDIV3 to set the clocks, the clock for the timers (FTM0, FTM2, PWT) must not be slower than bus clock.

Address: 4004_8000h base + 1Ch offset = 4004_801Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		OUTDIV1		0			OUTDIV2	0			OUTDIV3	0			
W	▨	▨			▨	▨	▨		▨	▨	▨		▨	▨	▨	▨
Reset	0	0	u*	u*	0	0	0	u*	0	0	0	u*	0	0	0	0
POR/ LVD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
POR/ LVD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- u = Unaffected by reset.

SIM_CLKDIV field descriptions

Field	Description
31–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–28 OUTDIV1	Clock 1 output divider value This field sets the divide value for the core/system clock,. 00 Same as ICSOUTCLK. 01 ICSOUTCLK divides by 2. 10 ICSOUTCLK divides by 3. 11 ICSOUTCLK divides by 4.

Table continues on the next page...

SIM_CLKDIV field descriptions (continued)

Field	Description
27–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 OUTDIV2	Clock 2 output divider value This field sets the divide value for the bus/FLASH, follow OUTDIV1. 0 Not divided from divider1. 1 Divide by 2 from divider1.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 OUTDIV3	Clock 3 output divider value This field sets the divide value for the timers(FTM0, FTM2,PWT). 0 Same as ICSOUTCLK. 1 ICSOUTCLK divides by 2.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

12.3 Functional description

See Introduction section.

Chapter 13
Power Management Controller (PMC)

13.1 Introduction

This chapter describes the functionality of the individual modules in the chip's low-power modes and the operation of Power Management Controller module.

13.2 Low voltage detect (LVD) system

This device includes a system to protect against low voltage conditions in order to protect memory contents and control MCU system states during supply voltage variations. This system consists of a power-on reset (POR) circuit and an LVD circuit with a user selectable trip voltage, either high (V_{LVDH}) or low (V_{LVDL}). The LVD circuit is enabled when SPMSC1[LVDE] is set and the trip voltage is selected by SPMSC2[LVDV]. The LVD is disabled upon entering the Stop mode unless SPMSC1[LVDSE] is set. If SPMSC1[LVDSE] and SPMSC1[LVDE] are both set, the current consumption will be greater in Stop mode with the LVD system enabled.

The following figure presents the block diagram of the low-voltage detect (LVD) system.

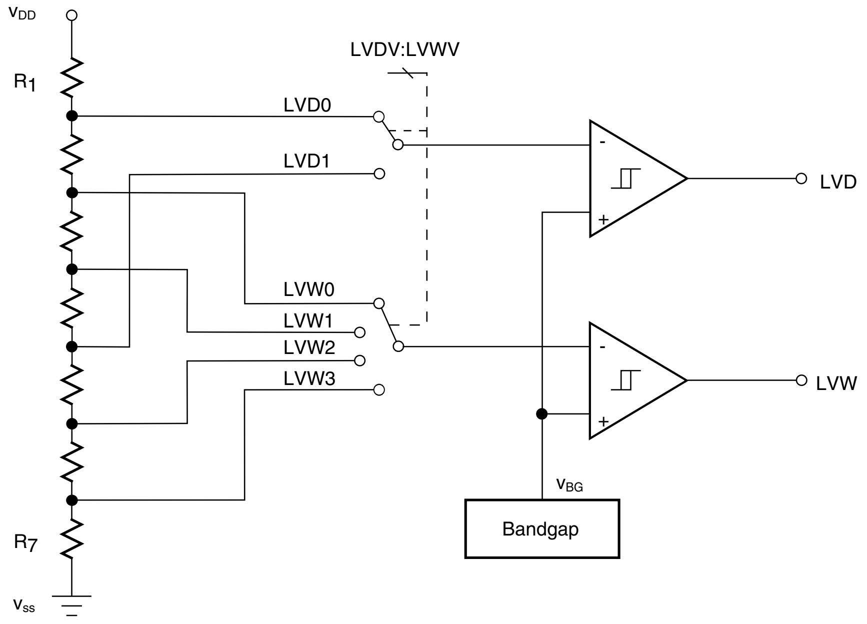


Figure 13-1. Low voltage detect (LVD) block diagram

13.2.1 Power-on reset (POR) operation

When power is initially applied to the MCU, or when the supply voltage drops below the V_{POR} level, the POR circuit will cause a reset condition. As the supply voltage rises, the LVD circuit will hold the chip in reset until the supply has risen above the V_{LVDL} level. Both the SIM_SRSID[POR] and SIM_SRSID[LVD] are set following a POR.

13.2.2 LVD reset operation

The LVD can be configured to generate a reset upon detection of a low-voltage condition by setting SPMSC1[LVDRE] to 1. After an LVD reset has occurred, the LVD system will hold the MCU in reset until the supply voltage has risen above the level determined by LVDV. SIM_SRSID[LVD] is set following either an LVD reset or POR.

13.2.3 LVD enabled in Stop mode

The LVD system is capable of generating a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in Stop (both SPMSC1[LVDE] and SPMSC1[LVDSE] set to 1) at the time the CPU executes a STOP instruction, then the voltage regulator remains active during Stop mode.

13.2.4 Low-voltage warning (LVW)

The LVD system has a low voltage warning flag to indicate that the supply voltage is approaching the LVW voltage. When a low voltage condition is detected and the LVD circuit is configured for interrupt operation (SPMSC1[LVDE] set, SPMSC1[LVWIE] set), SPMSC1[LVWF] will be set and LVW interrupt will occur. There are four user-selectable trip voltages for the LVW upon each LVDV configuration. The trip voltage is selected by SPMSC2[LVWV].

13.3 Bandgap reference

This device includes an on-chip bandgap reference (≈1.2 V) connected to the ADC channel. The bandgap reference voltage will not drop under the full operating voltage even when the operating voltage is falling. This reference voltage acts as an ideal reference voltage for accurate measurements.

13.4 Memory map and register descriptions

PMC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_D000	System Power Management Status and Control 1 Register (PMC_SPMSC1)	8	R/W	1Ch	13.4.1/174
4007_D001	System Power Management Status and Control 2 Register (PMC_SPMSC2)	8	R/W	00h	13.4.2/175

13.4.1 System Power Management Status and Control 1 Register (PMC_SPMSC1)

This high-page register contains status and control bits to support the low-voltage detection function, and to enable the bandgap voltage reference for use by the ADC module. This register must be written during the user's reset initialization program to set the desired controls, even if the desired settings are the same as the reset settings.

Address: 4007_D000h base + 0h offset = 4007_D000h

Bit	7	6	5	4	3	2	1	0
Read	LVWF	0	LVWIE	LVDRE	LVDSE	LVDE	▨	BGBE
Write	▨	LVWACK					0	
Reset	0	0	0	1	1	1	0	0

PMC_SPMSC1 field descriptions

Field	Description
7 LVWF	Low-Voltage Warning Flag Indicates the low-voltage warning status. **NOTE:** LVWF will be set in the case when V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVW}. LVWF may be 1 after power-on-reset, therefore, to use LVW interrupt function, before enabling LVWIE, LVWF must be cleared by writing LVWACK first. 0 Low-voltage warning is not present. 1 Low-voltage warning is present or was present.
6 LVWACK	Low-Voltage Warning Acknowledge If LVWF = 1, a low-voltage condition has occurred. To acknowledge this low-voltage warning, write 1 to LVWACK, which automatically clears LVWF to 0 if the low-voltage warning is no longer present.
5 LVWIE	Low-Voltage Warning Interrupt Enable Enables hardware interrupt requests for LVWF. 0 Hardware interrupt is disabled (use polling). 1 Requests a hardware interrupt when LVWF = 1.
4 LVDRE	Low-Voltage Detect Reset Enable This write-once bit enables LVD events to generate a hardware reset (provided LVDE = 1). **NOTE:** This field can be written only one time after reset. Additional writes are ignored. If LVDRE = 0, use LVW to monitor status because no flag was assert. 0 LVD events do not generate hardware resets. 1 Forces an MCU reset when an enabled low-voltage detect event occurs.
3 LVDSE	Low-Voltage Detect Stop Enable

Table continues on the next page...

PMC_SPMSC1 field descriptions (continued)

Field	Description
	Provided LVDE = 1, this read/write field determines whether the low-voltage detect function operates when the MCU is in Stop mode. 0 Low-voltage detect is disabled during Stop mode. 1 Low-voltage detect is enabled during Stop mode.
2 LVDE	Low-Voltage Detect Enable This write-once bit enables low-voltage detect logic and qualifies the operation of other fields in this register. **NOTE:** This field can be written only one time after reset. Additional writes are ignored. 0 LVD logic is disabled. 1 LVD logic is enabled.
1 Reserved	This field is reserved.
0 BGBE	Bandgap Buffer Enable Enables an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels or bandgap selected as ACMP's reference. 0 Bandgap buffer is disabled. 1 Bandgap buffer is enabled.

13.4.2 System Power Management Status and Control 2 Register (PMC_SPMSC2)

This register is used to report the status of the low-voltage warning function, and to configure the Stop mode behavior of the MCU. This register should be written during the user's reset initialization program to set the desired controls, even if the desired settings are the same as the reset settings.

Address: 4007_D000h base + 1h offset = 4007_D001h

Bit	7	6	5	4	3	2	1	0
Read	0	LVDV	LVWV		0			
Write	▨				▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

PMC_SPMSC2 field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 LVDV	Low-Voltage Detect Voltage Select This write-once bit selects the low-voltage detect (LVD) trip point setting. See data sheet for details.

Table continues on the next page...

PMC_SPMSC2 field descriptions (continued)

Field	Description
	0 Low trip point is selected ($V_{LVD} = V_{LVDL}$). 1 High trip point is selected ($V_{LVD} = V_{LVDH}$).
5–4 LVWV	Low-Voltage Warning Voltage Select Selects the low-voltage warning (LVW) trip point voltage. See data sheet for details. 00 Low trip point is selected ($V_{LVW} = V_{LVW1}$). 01 Middle 1 trip point is selected ($V_{LVW} = V_{LVW2}$). 10 Middle 2 trip point is selected ($V_{LVW} = V_{LVW3}$). 11 High trip point is selected ($V_{LVW} = V_{LVW4}$).
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Chapter 14
Miscellaneous Control Module (MCM)

14.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The Miscellaneous Control Module (MCM) provides a myriad of miscellaneous control functions.

14.1.1 Features

The MCM includes the following features:

- Program-visible information on the platform configuration
- Flash controller speculation buffer and cache configurations

14.2 Memory map/register descriptions

The memory map and register descriptions found here describe the registers using byte addresses. The registers can be written only when in supervisor mode.

MCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F000_3008	Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)	16	R	0007h	14.2.1/178
F000_300A	Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)	16	R	0001h	14.2.2/178
F000_300C	Platform Control Register (MCM_PLACR)	32	R/W	0000_0800h	14.2.3/179

14.2.1 Crossbar Switch (AXBS) Slave Configuration (MCM_PLASC)

PLASC is a 16-bit read-only register identifying the presence/absence of bus slave connections to the device’s crossbar switch.

Address: F000_3000h base + 8h offset = F000_3008h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								ASC							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

MCM_PLASC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ASC	Each bit in the ASC field indicates whether there is a corresponding connection to the crossbar switch's slave input port. 0 A bus slave connection to AXBS input port *n* is absent. 1 A bus slave connection to AXBS input port *n* is present.

14.2.2 Crossbar Switch (AXBS) Master Configuration (MCM_PLAMC)

PLAMC is a 16-bit read-only register identifying the presence/absence of bus master connections to the device's crossbar switch.

Address: F000_3000h base + Ah offset = F000_300Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0								AMC							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MCM_PLAMC field descriptions

Field	Description
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
AMC	Each bit in the AMC field indicates whether there is a corresponding connection to the AXBS master input port.

Table continues on the next page...

MCM_PLAMC field descriptions (continued)

Field	Description
	0 A bus master connection to AXBS input port *n* is absent 1 A bus master connection to AXBS input port *n* is present

14.2.3 Platform Control Register (MCM_PLACR)

The speculation buffer and cache in the flash memory controller is configurable via PLACR[15:10].

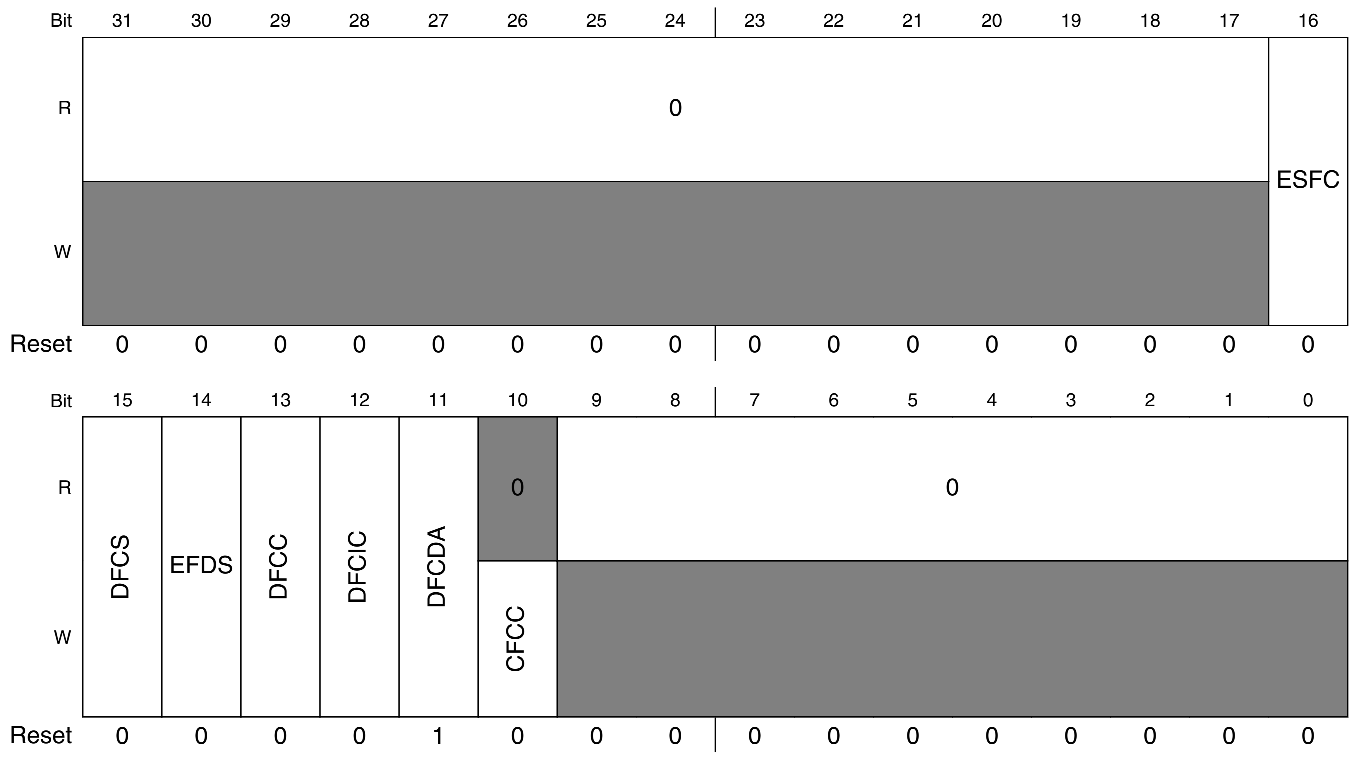
The speculation buffer is enabled only for instructions after reset. It is possible to have these states for the speculation buffer:

DFCS	EFDS	Description
0	0	Speculation buffer is on for instruction and off for data.
0	1	Speculation buffer is on for instruction and on for data.
1	X	Speculation buffer is off.

The cache in flash controller is enabled and caching both instruction and data type fetches after reset. It is possible to have these states for the cache:

DFCC	DFCIC	DFCDA	Description
0	0	0	Cache is on for both instruction and data.
0	0	1	Cache is on for instruction and off for data.
0	1	0	Cache is off for instruction and on for data.
0	1	1	Cache is off for both instruction and data.
1	X	X	Cache is off.

Address: F000_3000h base + Ch offset = F000_300Ch



MCM_PLACR field descriptions

Field	Description
31–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ESFC	Enable Stalling Flash Controller Enables stalling flash controller when flash is busy. When software needs to access the flash memory while a flash memory resource is being manipulated by a flash command, software can enable a stall mechanism to avoid a read collision. The stall mechanism allows software to execute code from the same block on which flash operations are being performed. However, software must ensure the sector the flash operations are being performed on is not the same sector from which the code is executing. ESFC enables the stall mechanism. This bit must be set only just before the flash operation is executed and must be cleared when the operation completes. 0 Disable stalling flash controller when flash is busy. 1 Enable stalling flash controller when flash is busy.
15 DFCS	Disable Flash Controller Speculation Disables flash controller speculation. 0 Enable flash controller speculation. 1 Disable flash controller speculation.
14 EFDS	Enable Flash Data Speculation Enables flash data speculation.

Table continues on the next page...

MCM_PLACR field descriptions (continued)

Field	Description
	0 Disable flash data speculation. 1 Enable flash data speculation.
13 DFCC	Disable Flash Controller Cache Disables flash controller cache. 0 Enable flash controller cache. 1 Disable flash controller cache.
12 DFCIC	Disable Flash Controller Instruction Caching Disables flash controller instruction caching. 0 Enable flash controller instruction caching. 1 Disable flash controller instruction caching.
11 DFCDA	Disable Flash Controller Data Caching Disables flash controller data caching. 0 Enable flash controller data caching 1 Disable flash controller data caching.
10 CFCC	Clear Flash Controller Cache Writing a 1 to this field clears the cache. Writing a 0 to this field is ignored. This field always reads as 0.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Chapter 15
Peripheral Bridge (AIPS-Lite)

15.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The peripheral bridge converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

The peripheral bridge occupies 64 MB of the address space, which is divided into peripheral slots of 4 KB. (It might be possible that all the peripheral slots are not used. See the memory map chapter for details on slot assignments.) The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

15.1.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width

15.1.2 General operation

The slave devices connected to the peripheral bridge are modules which contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge. The peripheral bridge performs a bus protocol conversion of the master transactions and generates the following as inputs to the peripherals:

- Module enables
- Module addresses

- Transfer attributes
- Byte enables
- Write data

The peripheral bridge selects and captures read data from the peripheral interface and returns it to the crossbar switch.

The register maps of the peripherals are located on 4-KB boundaries. Each peripheral is allocated one or more 4-KB block(s) of the memory map.

The AIPS-Lite module uses the data width of accessed peripheral to perform proper data byte lane routing; bus decomposition (bus sizing) is performed when the access size is larger than the peripheral's data width.

15.2 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus.

The peripheral bridge manages all transactions destined for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

15.2.1 Access support

All combinations of access size and peripheral data port width are supported. An access that is larger than the target peripheral's data width will be decomposed to multiple, smaller accesses. Bus decomposition is terminated by a transfer error caused by an access to an empty register area.

Chapter 16
Watchdog Timer (WDOG)

16.1 Introduction

The Watchdog Timer (WDOG) module is an independent timer that is available for system use. It provides a safety feature to ensure that software is executing as planned and that the CPU is not stuck in an infinite loop or executing unintended code. If the WDOG module is not serviced (refreshed) within a certain period, it resets the MCU.

16.1.1 Features

Features of the WDOG module include:

- Configurable clock source inputs independent from the bus clock
    - Internal 32 kHz RC oscillator
    - Internal 1 kHz RC oscillator
    - External clock source
- Programmable timeout period
    - Programmable 16-bit timeout value
    - Optional fixed 256 clock prescaler when longer timeout periods are needed
- Robust write sequence for counter refresh
    - Refresh sequence of writing 0x02A6 and then 0x80B4 within 16 bus clocks
- Window mode option for the refresh mechanism
    - Programmable 16-bit window value

- Provides robust check that program flow is faster than expected
- Early refresh attempts trigger a reset.

- Optional timeout interrupt to allow post-processing diagnostics
    - Interrupt request to CPU with interrupt vector for an interrupt service routine (ISR)
    - Forced reset occurs 128 bus clocks after the interrupt vector fetch.
- Configuration bits are write-once-after-reset to ensure watchdog configuration cannot be mistakenly altered.
- Robust write sequence for unlocking write-once configuration bits
    - Unlock sequence of writing 0x20C5 and then 0x28D9 within 16 bus clocks for allowing updates to write-once configuration bits
    - Software must make updates within 128 bus clocks after unlocking and before WDOG closing unlock window.

16.1.2 Block diagram

The following figure provides a block diagram of the WDOG module.

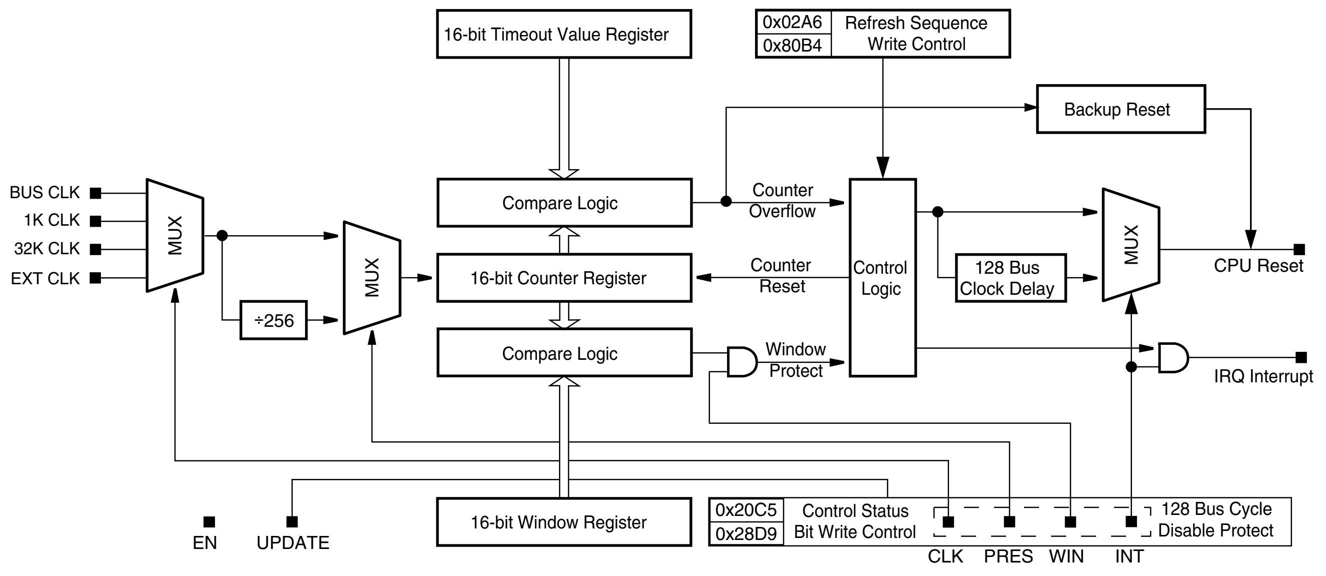


Figure 16-1. WDOG block diagram

16.2 Memory map and register definition

NOTE

If the device uses half-word to access WDOG_CNT, WDOG_TOVAL and WDOG_WIN, the transposed 16-bit bytes must follow the format of LowByte:HighByte. So 8-bit R/W is preferred.

WDOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4005_2000	Watchdog Control and Status Register 1 (WDOG_CS1)	8	R/W	80h	16.2.1/187
4005_2001	Watchdog Control and Status Register 2 (WDOG_CS2)	8	R/W	01h	16.2.2/189
4005_2002	Watchdog Counter Register: High (WDOG_CNTH)	8	R	00h	16.2.3/190
4005_2003	Watchdog Counter Register: Low (WDOG_CNTL)	8	R	00h	16.2.4/190
4005_2004	Watchdog Timeout Value Register: High (WDOG_TOVALH)	8	R/W	00h	16.2.5/191
4005_2005	Watchdog Timeout Value Register: Low (WDOG_TOVALL)	8	R/W	04h	16.2.6/191
4005_2006	Watchdog Window Register: High (WDOG_WINH)	8	R/W	00h	16.2.7/192
4005_2007	Watchdog Window Register: Low (WDOG_WINL)	8	R/W	00h	16.2.8/192

16.2.1 Watchdog Control and Status Register 1 (WDOG_CS1)

This section describes the function of Watchdog Control and Status Register 1.

NOTE

TST is cleared (0:0) on POR only. Any other reset does not affect the value of this field.

Address: 4005_2000h base + 0h offset = 4005_2000h

Bit	7	6	5	4	3	2	1	0
Read Write	EN	INT	UPDATE	TST		DBG	WAIT	STOP
Reset	1	0	0	0	0	0	0	0

WDOG_CS1 field descriptions

Field	Description
7 EN	Watchdog Enable This write-once bit enables the watchdog counter to start counting. 0 Watchdog disabled. 1 Watchdog enabled.

Table continues on the next page...

WDOG_CS1 field descriptions (continued)

Field	Description
6 INT	Watchdog Interrupt This write-once bit configures the watchdog to generate an interrupt request upon a reset-triggering event (timeout or illegal write to the watchdog), prior to forcing a reset. After the interrupt vector fetch, the reset occurs after a delay of 128 bus clocks. 0 Watchdog interrupts are disabled. Watchdog resets are not delayed. 1 Watchdog interrupts are enabled. Watchdog resets are delayed by 128 bus clocks.
5 UPDATE	Allow updates This write-once bit allows software to reconfigure the watchdog without a reset. 0 Updates not allowed. After the initial configuration, the watchdog cannot be later modified without forcing a reset. 1 Updates allowed. Software can modify the watchdog configuration registers within 128 bus clocks after performing the unlock write sequence.
4–3 TST	Watchdog Test Enables the fast test mode. The test mode allows software to exercise all bits of the counter to demonstrate that the watchdog is functioning properly. See the Fast testing of the watchdog section. This write-once field is cleared (0:0) on POR only. Any other reset does not affect the value of this field. 00 Watchdog test mode disabled. 01 Watchdog user mode enabled. (Watchdog test mode disabled.) After testing the watchdog, software should use this setting to indicate that the watchdog is functioning normally in user mode. 10 Watchdog test mode enabled, only the low byte is used. WDOG_CNTL is compared with WDOG_TOVALL. 11 Watchdog test mode enabled, only the high byte is used. WDOG_CNTH is compared with WDOG_TOVALH.
2 DBG	Debug Enable This write-once bit enables the watchdog to operate when the chip is in debug mode. 0 Watchdog disabled in chip debug mode. 1 Watchdog enabled in chip debug mode.
1 WAIT	Wait Enable This write-once bit enables the watchdog to operate when the chip is in wait mode. 0 Watchdog disabled in chip wait mode. 1 Watchdog enabled in chip wait mode.
0 STOP	Stop Enable This write-once bit enables the watchdog to operate when the chip is in stop mode. 0 Watchdog disabled in chip stop mode. 1 Watchdog enabled in chip stop mode.

16.2.2 Watchdog Control and Status Register 2 (WDOG_CS2)

This section describes the function of the watchdog control and status register 2.

Address: 4005_2000h base + 1h offset = 4005_2001h

Bit	7	6	5	4	3	2	1	0
Read	WIN	FLG	0	PRES	0		CLK	
Write		w1c	▨		▨	▨		
Reset	0	0	0	0	0	0	0	1

WDOG_CS2 field descriptions

Field	Description
7 WIN	Watchdog Window This write-once bit enables window mode. See the Window mode section. 0 Window mode disabled. 1 Window mode enabled.
6 FLG	Watchdog Interrupt Flag This bit is an interrupt indicator when INT is set in control and status register 1. Write 1 to clear it. 0 No interrupt occurred. 1 An interrupt occurred.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 PRES	Watchdog Prescalar This write-once bit enables a fixed 256 pre-scaling of watchdog counter reference clock. (The block diagram shows this clock divider option.) 0 256 prescalar disabled. 1 256 prescalar enabled.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CLK	Watchdog Clock This write-once field indicates the clock source that feeds the watchdog counter. See the Clock source section. 00 Bus clock. 01 1 kHz internal low-power oscillator (LPOCLK). 10 32 kHz internal oscillator (ICSIRCLK). 11 External clock source.

16.2.3 Watchdog Counter Register: High (WDOG_CNTH)

This section describes the watchdog counter registers: high (CNTH) and low (CNTL) combined.

The watchdog counter registers CNTH and CNTL provide access to the value of the free-running watchdog counter. Software can read the counter registers at any time.

Software cannot write directly to the watchdog counter; however, two write sequences to these registers have special functions:

1. The *refresh sequence* resets the watchdog counter to 0x0000. See the Refreshing the Watchdog section.
2. The *unlock sequence* allows the watchdog to be reconfigured without forcing a reset (when WDOG_CS1[UPDATE] = 1). See the Example code: Reconfiguring the Watchdog section.

NOTE

All other writes to these registers are illegal and force a reset.

Address: 4005_2000h base + 2h offset = 4005_2002h

Bit	7	6	5	4	3	2	1	0
Read	CNTHIGH							
Write	▨							
Reset	0	0	0	0	0	0	0	0

WDOG_CNTH field descriptions

Field	Description
CNTHIGH	High byte of the Watchdog Counter

16.2.4 Watchdog Counter Register: Low (WDOG_CNTL)

See the description of the WDOG_CNTH register.

Address: 4005_2000h base + 3h offset = 4005_2003h

Bit	7	6	5	4	3	2	1	0
Read	CNTLOW							
Write	▨							
Reset	0	0	0	0	0	0	0	0

WDOG_CNTL field descriptions

Field	Description
CNTLOW	Low byte of the Watchdog Counter

16.2.5 Watchdog Timeout Value Register: High (WDOG_TOVALH)

This section describes the watchdog timeout value registers: high (WDOG_TOVALH) and low (WDOG_TOVALL) combined. WDOG_TOVALH and WDOG_TOVALL contains the 16-bit value used to set the timeout period of the watchdog.

The watchdog counter (WDOG_CNTH and WDOG_CNTL) is continuously compared with the timeout value (WDOG_TOVALH and WDOG_TOVALL). If the counter reaches the timeout value, the watchdog forces a reset.

NOTE

Do not write 0 to the Watchdog Timeout Value Register, otherwise, the watchdog always generates a reset.

Address: 4005_2000h base + 4h offset = 4005_2004h

Bit	7	6	5	4	3	2	1	0
Read Write	TOVALHIGH							
Reset	0	0	0	0	0	0	0	0

WDOG_TOVALH field descriptions

Field	Description
TOVALHIGH	High byte of the timeout value

16.2.6 Watchdog Timeout Value Register: Low (WDOG_TOVALL)

See the description of the WDOG_TOVALH register.

NOTE

All the bits reset to 0 in read.

Address: 4005_2000h base + 5h offset = 4005_2005h

Bit	7	6	5	4	3	2	1	0
Read Write	TOVALLOW							
Reset	0	0	0	0	0	1	0	0

WDOG_TOVALL field descriptions

Field	Description
TOVALLOW	Low byte of the timeout value

16.2.7 Watchdog Window Register: High (WDOG_WINH)

This section describes the watchdog window registers: high (WDOG_WINH) and low (WDOG_WINL) combined. When window mode is enabled (WDOG_CS2[WIN] is set), WDOG_WINH and WDOG_WINL determine the earliest time that a refresh sequence is considered valid. See the Watchdog refresh mechanism section.

WDOG_WINH and WDOG_WINL must be less than WDOG_TOVALH and WDOG_TOVALL.

Address: 4005_2000h base + 6h offset = 4005_2006h

Bit	7	6	5	4	3	2	1	0
Read Write	WINHIGH							
Reset	0	0	0	0	0	0	0	0

WDOG_WINH field descriptions

Field	Description
WINHIGH	High byte of Watchdog Window

16.2.8 Watchdog Window Register: Low (WDOG_WINL)

See the description of the WDOG_WINH register.

Address: 4005_2000h base + 7h offset = 4005_2007h

Bit	7	6	5	4	3	2	1	0
Read Write	WINLOW							
Reset	0	0	0	0	0	0	0	0

WDOG_WINL field descriptions

Field	Description
WINLOW	Low byte of Watchdog Window

16.3 Functional description

The WDOG module provides a fail safe mechanism to ensure the system can be reset to a known state of operation in case of system failure, such as the CPU clock stopping or there being a run away condition in the software code. The watchdog counter runs continuously off a selectable clock source and expects to be serviced (refreshed) periodically. If it is not, it resets the system.

The timeout period, window mode, and clock source are all programmable but must be configured within 128 bus clocks after a reset.

16.3.1 Watchdog refresh mechanism

The watchdog resets the MCU if the watchdog counter is not refreshed. A robust refresh mechanism makes it very unlikely that the watchdog can be refreshed by runaway code.

To refresh the watchdog counter, software must execute a refresh write sequence before the timeout period expires. In addition, if window mode is used, software must not start the refresh sequence until after the time value set in the WDOG_WINH and WDOG_WINL registers. See the following figure.

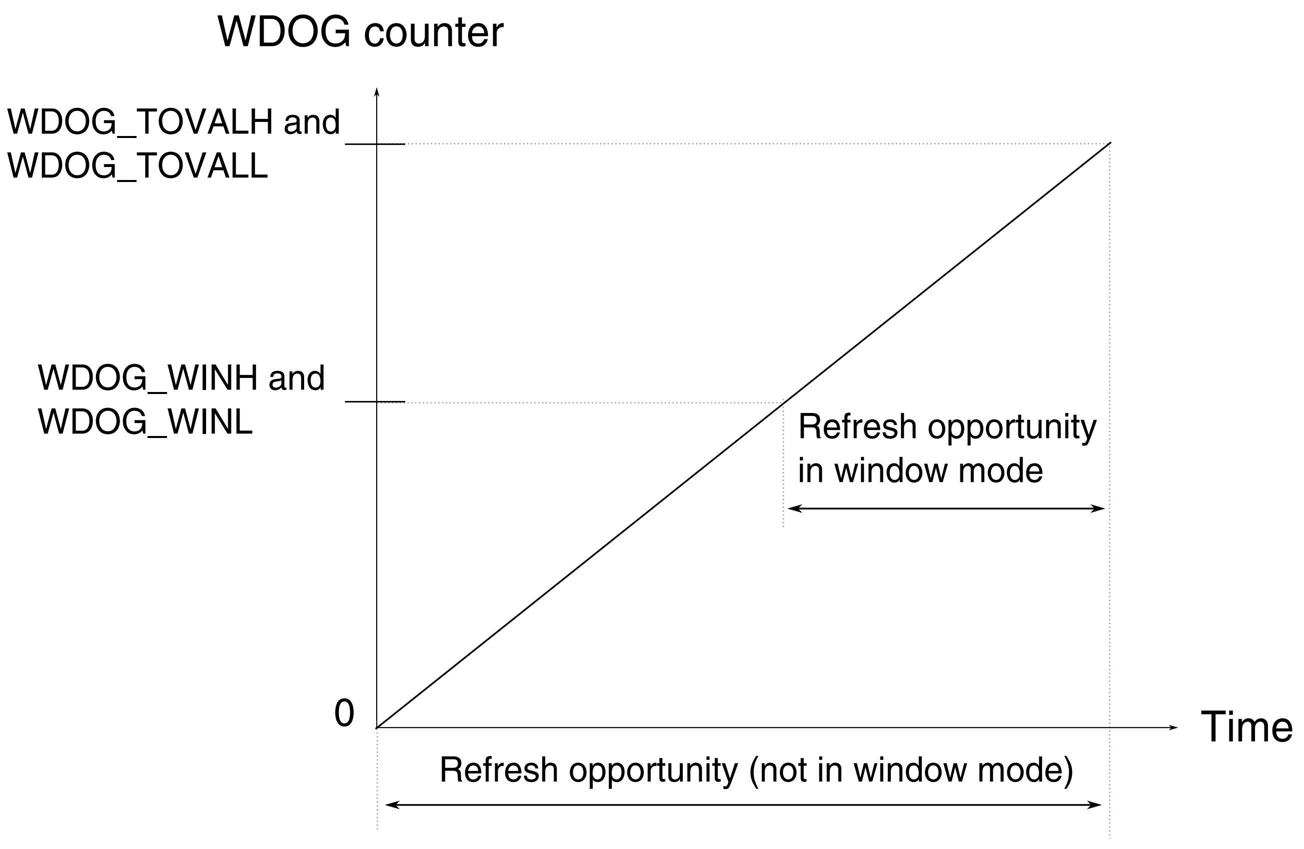

WDOG counter
WDOG_TOVALH and WDOG_TOVALL
WDOG_WINH and WDOG_WINL
Refresh opportunity in window mode
0
Time
Refresh opportunity (not in window mode)

Figure 16-10. Refresh opportunity for the Watchdog counter

16.3.1.1 Window mode

Software finishing its main control loop faster than expected could be an indication of a problem. Depending on the requirements of the application, the WDOG can be programmed to force a reset when refresh attempts are early.

When Window mode is enabled, the watchdog must be refreshed after the counter has reached a minimum expected time value; otherwise, the watchdog resets the MCU. The minimum expected time value is specified in the WDOG_WINH:L registers. Setting CS1[WIN] enables Window mode.

16.3.1.2 Refreshing the Watchdog

The refresh write sequence is a write of 0x02A6 followed by a write of 0x80B4 to the WDOG_CNTH and WDOG_CNTL registers. The write of the 0x80B4 must occur within 16 bus clocks after the write of 0x02A6; otherwise, the watchdog resets the MCU.

Note

Before starting the refresh sequence, disable global interrupts. Otherwise, an interrupt could effectively invalidate the refresh sequence if writing the four bytes takes more than 16 bus clocks. Re-enable interrupts when the sequence is finished.

16.3.1.3 Example code: Refreshing the Watchdog

The following code segment shows the refresh write sequence of the WDOG module.

NOTE

The following example code combines the 8-bit WDOG_CNTH and WDOG_CNTL as one 16-bit WDOG_CNT, the 8-bit WDOG_TOVALH and WDOG_TOVALL as one 16-bit WDOG_TOVAL, WDOG_WINH and WDOG_WINL as WDOG_WIN and uses 16-bit access.

```
/* Refresh watchdog */

for (;;) // main loop
{
   ...

   DisableInterrupts; // disable global interrupt

   WDOG_CNT = 0x02A6; // write the 1st refresh word

   WDOG_CNT = 0x80B4; // write the 2nd refresh word to refresh counter

   EnableInterrupts; // enable global interrupt

   ...
}
```

16.3.2 Configuring the Watchdog

All watchdog control bits, timeout value, and window value are write-once after reset. This means that after a write has occurred they cannot be changed unless a reset occurs. This provides a robust mechanism to configure the watchdog and ensure that a runaway condition cannot mistakenly disable or modify the watchdog configuration after configured.

This is guaranteed by the user configuring the window and timeout value first, followed by the other control bits, and ensuring that CS1[UPDATE] is also set to 0. The new configuration takes effect only after all registers except WDOG_CNTH:L are written once after reset. Otherwise, the WDOG uses the reset values by default. If window mode is not used (CS2[WIN] is 0), writing to WDOG_WINH:L is not required to make the new configuration take effect.

16.3.2.1 Reconfiguring the Watchdog

In some cases (such as when supporting a bootloader function), users may want to reconfigure or disable the watchdog without forcing a reset first. By setting CS1[UPDATE] to a 1 on the initial configuration of the watchdog after a reset, users can reconfigure the watchdog at any time by executing an unlock sequence. (Conversely, if CS1[UPDATE] remains 0, the only way to reconfigure the watchdog is by initiating a reset.) The unlock sequence is similar to the refresh sequence but uses different values.

16.3.2.2 Unlocking the Watchdog

The unlock sequence is a write to the WDOG_CNTH:L registers of 0x20C5 followed by 0x28D9 within 16 bus clocks at any time after the watchdog has been configured. On completing the unlock sequence, the user must reconfigure the watchdog within 128 bus clocks; otherwise, the watchdog forces a reset to the MCU.

NOTE

Due to 128 bus clocks requirement for reconfiguring the watchdog, some delays must be inserted before executing STOP or WAIT instructions after reconfiguring the watchdog. This ensures that the watchdog's new configuration takes effect before MCU enters low power mode. Otherwise, the MCU may not be waken up from low power mode.

16.3.2.3 Example code: Reconfiguring the Watchdog

The following code segment shows an example reconfiguration of the WDOG module.

```
/* Initialize watchdog with ~1-kHz clock source, ~1s time-out */

DisableInterrupts; // disable global interrupt

WDOG_CNT = 0x20C5; // write the 1st unlock word

WDOG_CNT = 0x28D9; // write the 2nd unlock word

WDOG_TOVAL = 1000; // setting timeout value

WDOG_CS2 = WDOG_CS2_CLK_MASK; // setting 1-kHz clock source

WDOG_CS1 = WDOG_CS1_EN_MASK; // enable counter running
```

```
EnableInterrupts; // enable global interrupt
```

16.3.3 Clock source

The watchdog counter has four clock source options selected by programming CS2[CLK]:

- bus clock
- internal Low-Power Oscillator (LPO) running at approximately 1 kHz (This is the default source.)
- internal 32 kHz clock
- external clock

The options allow software to select a clock source independent of the bus clock for applications that need to meet more robust safety requirements. Using a clock source other than the bus clock ensures that the watchdog counter continues to run if the bus clock is somehow halted; see Backup reset.

An optional fixed prescaler for all clock sources allows for longer timeout periods. When CS2[PRES] is set, the clock source is prescaled by 256 before clocking the watchdog counter.

The following table summarizes the different watchdog timeout periods available.

Table 16-10. Watchdog timeout availability

Reference clock	Prescaler	Watchdog time-out availability
Internal ~1 kHz (LPO)	Pass through	~1 ms–65.5 s[1]
	÷256	~256 ms–16,777 s
Internal ~32 kHz	Pass through	~31.25 µs–2.048 s
	÷256	~8 ms–524.3 s
1 MHz (from bus or external)	Pass through	1 µs–65.54 ms
	÷256	256 µs–16.777 s
20 MHz (from bus or external)	Pass through	50 ns–3.277 ms
	÷256	12.8 µs–838.8 ms

1. The default timeout value after reset is approximately 4 ms.

NOTE

When the programmer switches clock sources during reconfiguration, the watchdog hardware holds the counter at zero for 2.5 periods of the previous clock source and 2.5 periods of the new clock source after the configuration time period (128 bus clocks) ends. This delay ensures a smooth transition before restarting the counter with the new configuration.

16.3.4 Using interrupts to delay resets

When interrupts are enabled (CS1[INT] = 1), the watchdog first generates an interrupt request upon a reset triggering event (such as a counter timeout or invalid refresh attempt). The watchdog delays forcing a reset for 128 bus clocks to allow the interrupt service routine (ISR) to perform tasks, such as analyzing the stack to debug code.

When interrupts are disabled (CS1[INT] = 0), the watchdog does not delay forcing a reset.

16.3.5 Backup reset

NOTE

A clock source other than the bus clock must be used as the reference clock for the counter; otherwise, the backup reset function is not available.

The backup reset function is a safeguard feature that independently generates a reset in case the main WDOG logic loses its clock (the bus clock) and can no longer monitor the counter. If the watchdog counter overflows twice in succession (without an intervening reset), the backup reset function takes effect and generates a reset.

16.3.6 Functionality in debug and low-power modes

By default, the watchdog is not functional in Active Background mode, Wait mode, or Stop mode. However, the watchdog can remain functional in these modes as follows:

- For Active Background mode, set CS1[DBG]. (This way the watchdog is functional in Active Background mode even when the CPU is held by the Debug module.)
- For Wait mode, set CS1[WAIT].
- For Stop mode, set CS1[STOP].

NOTE

The watchdog can not generate interrupt in Stop mode even if CS1[STOP] is set and will not wake the MCU from Stop mode. It can generate reset during Stop mode.

For Active Background mode and Stop mode, in addition to the above configurations, a clock source other than the bus clock must be used as the reference clock for the counter; otherwise, the watchdog cannot function.

16.3.7 Fast testing of the watchdog

Before executing application code in safety critical applications, users are required to test that the watchdog works as expected and resets the MCU. Testing every bit of a 16-bit counter by letting it run to the overflow value takes a relatively long time (64 kHz clocks).

To help minimize the startup delay for application code after reset, the watchdog has a feature to test the watchdog more quickly by splitting the counter into its constituent byte-wide stages. The low and high bytes are run independently and tested for timeout against the corresponding byte of the timeout value register. (For complete coverage when testing the high byte of the counter, the test feature feeds the input clock via the 8th bit of the low byte, thus ensuring that the overflow connection from the low byte to the high byte is tested.)

Using this test feature reduces the test time to 512 clocks (not including overhead, such as user configuration and reset vector fetches). To further speed testing, use a faster clock (such as the bus clock) for the counter reference.

On a power-on reset, the POR bit in the system reset register is set, indicating the user should perform the WDOG fast test.

16.3.7.1 Testing each byte of the counter

The test procedure follows these steps:

1. Program the preferred watchdog timeout value in the WDOG_TOVALH and WDOG_TOVALL registers during the watchdog configuration time period.
2. Select a byte of the counter to test using the WDOG_CS1[TST] = 10b for the low byte; WDOG_CS1[TST] = 11b for the high byte.
3. Wait for the watchdog to timeout. Optionally, in the idle loop, increment RAM locations as a parallel software counter for later comparison. Because the RAM is not affected by a watchdog reset, the timeout period of the watchdog counter can be compared with the software counter to verify the timeout period has occurred as expected.
4. The watchdog counter times out and forces a reset.

5. Confirm the WDOG flag in the system reset register is set, indicating that the watchdog caused the reset. (The POR flag remains clear.)
6. Confirm that WDOG_CS1[TST] shows a test (10b or 11b) was performed.

If confirmed, the count and compare functions work for the selected byte. Repeat the procedure, selecting the other byte in step 2.

NOTE

WDOG_CS1[TST] is cleared by a POR only and not affected by other resets.

16.3.7.2 Entering user mode

After successfully testing the low and high bytes of the watchdog counter, the user can configure WDOG_CS1[TST] to 01b to indicate the watchdog is ready for use in application user mode. Thus if a reset occurs again, software can recognize the reset trigger as a real watchdog reset caused by runaway or faulty application code.

As an ongoing test when using the default 1-kHz clock source, software can periodically read the WDOG_CNTH and WDOG_CNTL registers to ensure the counter is being incremented.

Chapter 17
Bit Manipulation Engine (BME)

17.1 Introduction

The Bit Manipulation Engine (BME) provides hardware support for atomic read-modify-write memory operations to the peripheral address space in Cortex-M0+ based microcontrollers.

This architectural capability is also known as "decorated storage" as it defines a mechanism for providing additional semantics for load and store operations to memory-mapped peripherals beyond just the reading and writing of data values to the addressed memory locations. In the BME definition, the "decoration", that is, the additional semantic information, is encoded into the peripheral address used to reference the memory.

By combining the basic load and store instructions of the ARM Cortex-M instruction set architecture (v6M, v7M) with the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. The resulting architectural capability defined by this core platform function is targeted at the manipulation of n-bit fields in peripheral registers and is consistent with I/O hardware addressing in the Embedded C standard. For most BME commands, a single core read or write bus cycle is converted into an atomic read-modify-write, that is, an indivisible "read followed by a write" bus sequence.

BME decorated references are only available on system bus transactions generated by the processor core and targeted at the standard 512 KB peripheral address space based at 0x4000_0000[1] and SRAM_U space based at 0x2000_0000. The decoration semantic is embedded into address bits[28:19], creating a 448 MB space at addresses 0x4400_0000–

1. To be perfectly accurate, the peripheral address space occupies a 516 KB region: 512 KB based at 0x4000_0000 plus a 4 KB space based at 0x400F_F000 for GPIO accesses. This organization provides compatibility with the Kinetis K Family. Attempted accesses to the memory space located between 0x4008_0000 - 0x400F_EFFF are error terminated due to an illegal address.

0x5FFF_FFFF for AIPS and a 448 MB space at addresses 0x2400_0000–0x3FFF_FFFF for SRAM_U; these bits are stripped out of the actual address sent to the peripheral bus controller and used by the BME to define and control its operation.

17.1.1 Overview

The following figure is a generic block diagram of the processor core and platform for this class of ultra low-end microcontrollers.

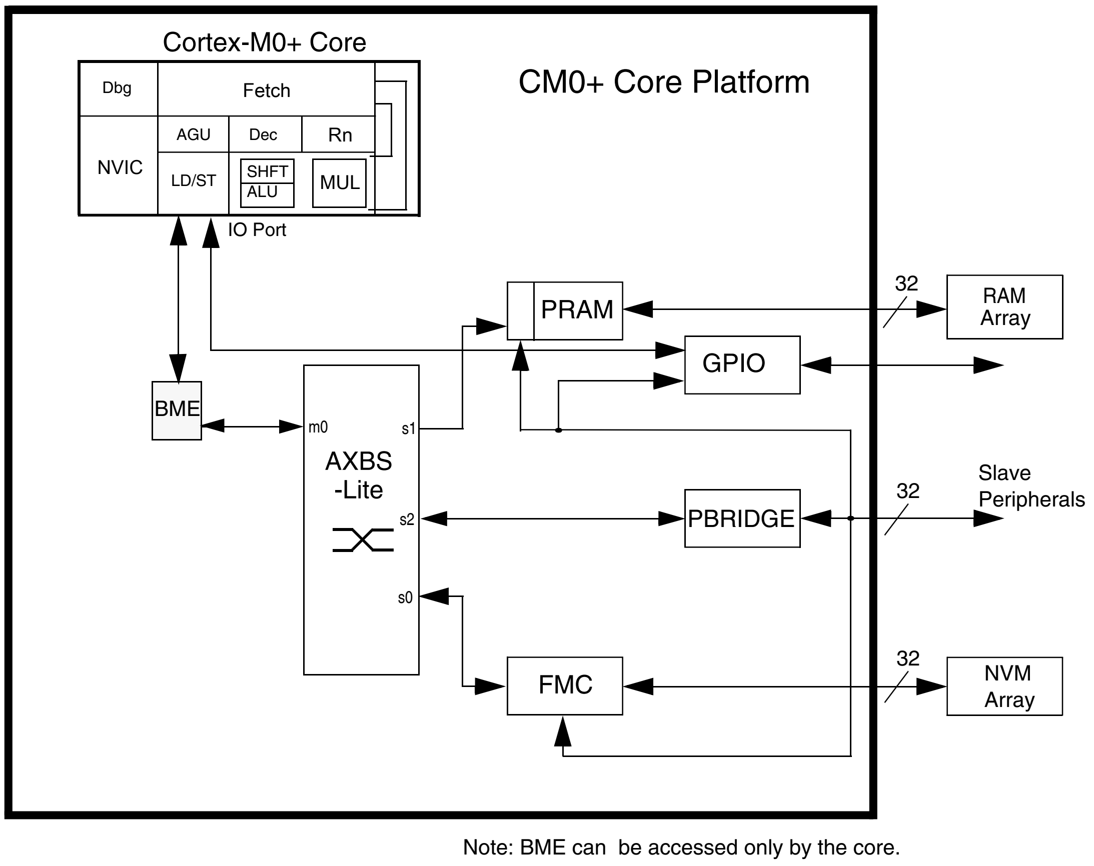


Figure 17-1. Cortex-M0+ core platform block diagram

As shown in the block diagram, the BME module interfaces to the master port on the crossbar switch allowing it to support atomic read-modify-write operations to the SRAM_U (shown as platform RAM (PRAM) in the figure) and the Peripheral Bridge (PBRIDGE) controller. The BME hardware microarchitecture is a 2-stage pipeline design matching the protocol of the AMBA-AHB system bus interfaces. The PBRIDGE module converts the AHB system bus protocol into the IPS/APB protocol used by the attached slave peripherals.

17.1.2 Features

The key features of the BME include:

- Lightweight implementation of decorated storage for selected address spaces
- Additional access semantics encoded into the reference address
- Resides between processor core and a switch master port
- Two-stage pipeline design matching the AHB system bus protocol
- Combinationally passes non-decorated accesses to slave bus controllers
- Conversion of decorated loads and stores from processor core into atomic read-modify-writes
- Decorated loads support unsigned bit field extracts, load-and-{set,clear} 1-bit operations
- Decorated stores support bit field inserts, logical AND, OR, and XOR operations
- Support for byte, halfword and word-sized decorated operations
- Supports minimum signal toggling on AHB output bus to reduce power dissipation

17.1.3 Modes of operation

The BME module does not support any special modes of operation. As a memory-mapped device located on a crossbar master AHB system bus port, BME responds strictly on the basis of memory addresses for accesses to the SRAM_U and peripheral bridge bus controller.

All functionality associated with the BME module resides in the core platform's clock domain; this includes its connections with the crossbar master port, SRAM_U and the PBRIDGE bus controller.

17.2 Memory map and register definition

The BME module provides a memory-mapped capability and does not include any programming model registers.

The exact set of functions supported by the BME are detailed in the Functional description.

The peripheral address space occupies a 516 KB region: 512 KB based at 0x4000_0000 plus a 4 KB space based at 0x400F_F000 for GPIO accesses; the decorated address space is mapped to the 448 MB region located at 0x4400_0000–0x5FFF_FFFF. The decorated address space associated with the SRAM_U is the 448 MB region mapped at 0x2400_0000 - 0x3FFF_FFFF.

17.3 Functional description

Information found here details the specific functions supported by the BME.

Recall the combination of the basic load and store instructions of the Cortex-M instruction set architecture (v6M, v7M) plus the concept of decorated storage provided by the BME, the resulting implementation provides a robust and efficient read-modify-write capability to this class of ultra low-end microcontrollers. The resulting architectural capability defined by this core platform function is targeted at the manipulation of n-bit fields in peripheral registers and RAM and is consistent with I/O hardware addressing in the Embedded C standard. For most BME commands, a single core read or write bus cycle is converted into an atomic read-modify-write, that is, an indivisible "read followed by a write" bus sequence.

Consider decorated store operations first, then decorated loads.

17.3.1 BME decorated stores

The functions supported by the BME's decorated stores include three logical operators (AND, OR, XOR) plus a bit field insert.

For all these operations, BME converts a single decorated AHB store transaction into a 2-cycle atomic read-modify-write sequence, where the combined read-modify operation is performed in the first AHB data phase, and then the write is performed in the second AHB data phase.

A generic timing diagram of a decorated store showing a peripheral bit field insert operation is shown as follows:

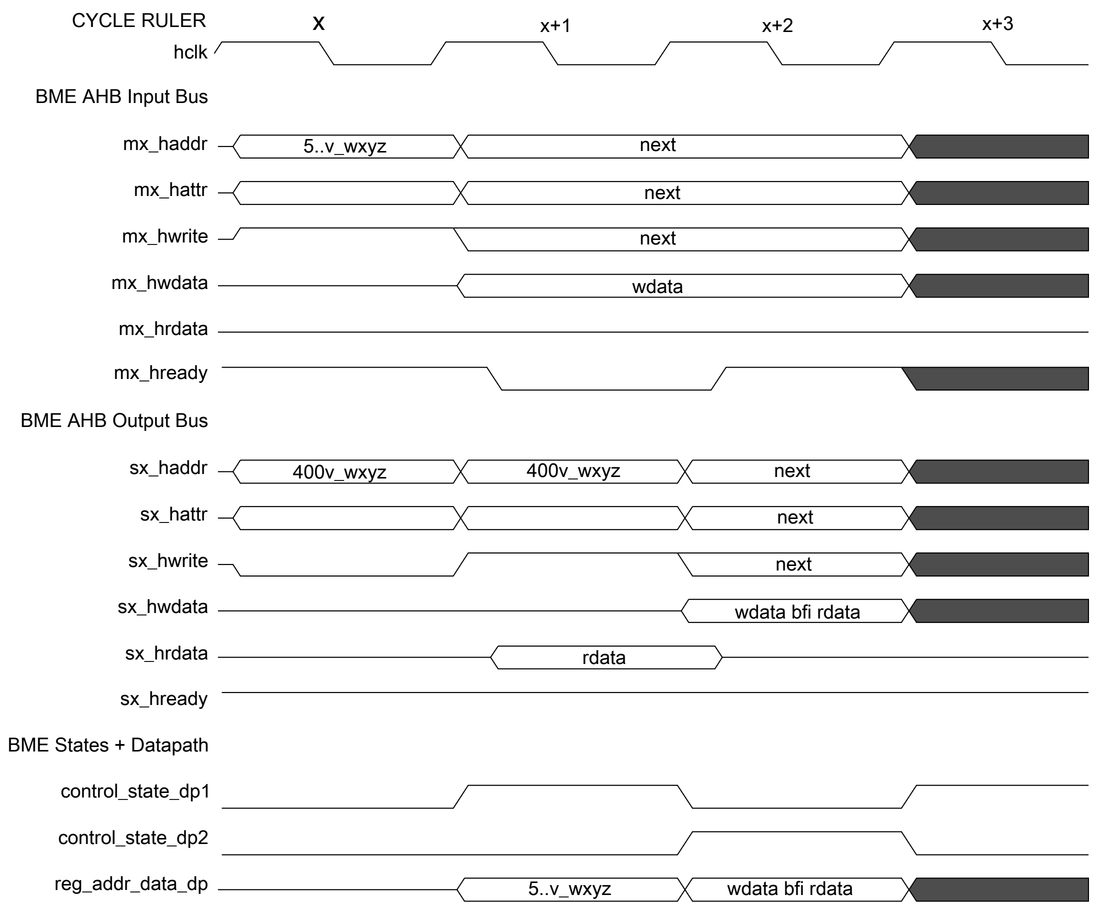


Figure 17-2. Decorated store: bit field insert timing diagram

All the decorated store operations follow the same execution template shown in Figure 17-2, a two-cycle read-modify-write operation:

1. Cycle x, 1st AHB address phase: Write from input bus is translated into a read operation on the output bus using the actual memory address (with the decoration removed) and then captured in a register.
2. Cycle x+1, 2nd AHB address phase: Write access with the registered (but actual) memory address is output
3. Cycle x+1, 1st AHB data phase: Memory read data is modified using the input bus write data and the function defined by the decoration and captured in a data register; the input bus cycle is stalled.
4. Cycle x+2, 2nd AHB data phase: Registered write data is sourced onto the output write data bus.

NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

17.3.1.1 Decorated store logical AND (AND)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read;
2. It is then modified by performing a logical AND operation using the write data operand sourced for the system bus cycle
3. Finally, the result of the AND operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

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

Figure 17-3. Decorated store address: logical AND

See Figure 17-3 where addr[30:29] = 01 for SRAM_U, addr[30:29] = 10 for peripheral, addr[28:26] = 001 specifies the AND operation, and mem_addr[19:0] specifies the address offset into the space based at 0x2000_0000 for SRAM_U, and 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care".

The decorated AND write operation is defined in the following pseudo-code as:

```
ioand<sz>(accessAddress, wdata)                    // decorated store AND
tmp   =  mem[accessAddress & 0xE00FFFFF, size]    // memory read
tmp   =  tmp & wdata                               // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp       // memory write
```

where the operand size <sz> is defined as b(yte, 8-bit), h(alfword, 16-bit) and w(ord, 32-bit). This notation is used throughout the document.

In the cycle definition tables, the notations AHB_ap and AHB_dp refer to the address and data phases of the BME AHB transaction. The cycle-by-cycle BME operations are detailed in the following table.

Table 17-1. Cycle definitions of decorated store: logical AND

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form (rdata & wdata) and capture destination data in register	Perform write sending registered data to memory

17.3.1.2 Decorated store logical OR (OR)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read.
2. It is then modified by performing a logical OR operation using the write data operand sourced for the system bus cycle.
3. Finally, the result of the OR operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

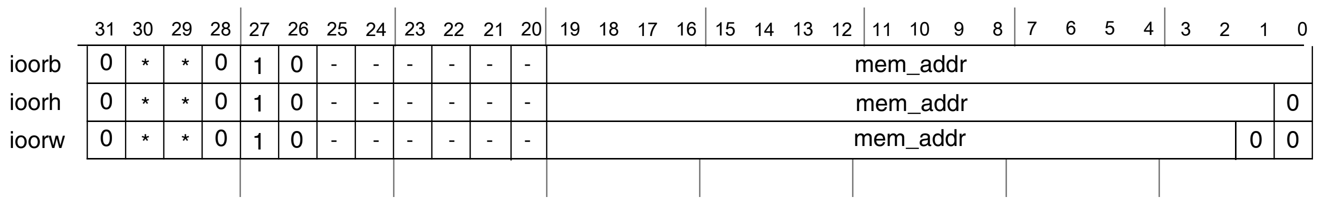

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ioorb 0 * * 0 1 0 - - - - - - mem_addr
ioorh 0 * * 0 1 0 - - - - - - mem_addr 0
ioorw 0 * * 0 1 0 - - - - - - mem_addr 0 0

Figure 17-4. Decorated address store: logical OR

See Figure 17-4,where addr[30:29] = 01 for SRAM_U, addr[30:29] =10 for peripheral, addr[28:26] = 010 specifies the OR operation, and mem_addr[19:0] specifies the address offset into the space based at 0x2000_0000 for SRAM_U, and 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care".

The decorated OR write operation is defined in the following pseudo-code as:

```
ioor<sz>(accessAddress, wdata)                  // decorated store OR

tmp   =  mem[accessAddress & 0xE00FFFFF, size]  // memory read
tmp   =  tmp | wdata                            // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp     // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 17-2. Cycle definitions of decorated store: logical OR

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form (rdata I wdata) and capture destination data in register	Perform write sending registered data to memory

17.3.1.3 Decorated store logical XOR (XOR)

This command performs an atomic read-modify-write of the referenced memory location.

1. First, the location is read.
2. It is then modified by performing a logical XOR (exclusive-OR) operation using the write data operand sourced for the system bus cycle.
3. Finally, the result of the XOR operation is written back into the referenced memory location.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). The core performs the required write data lane replication on byte and halfword transfers.

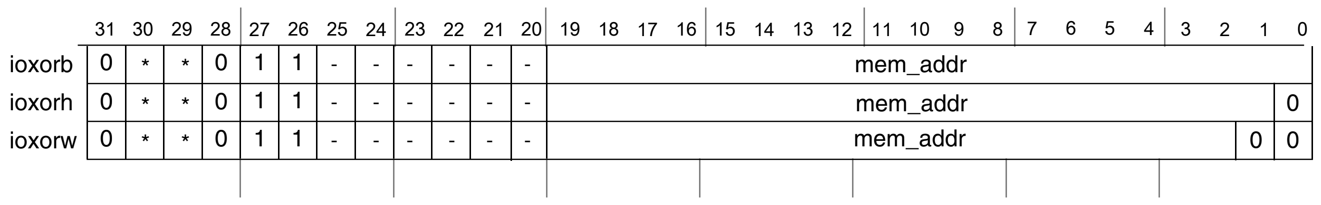

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ioxorb 0 * * 0 1 1 - - - - - - mem_addr
ioxorh 0 * * 0 1 1 - - - - - - mem_addr 0
ioxorw 0 * * 0 1 1 - - - - - - mem_addr 0 0

Figure 17-5. Decorated address store: logical XOR

See Figure 17-5, where addr[30:29] = 01 for SRAM_U, addr[30:29] =10 for peripheral, addr[28:26] = 011 specifies the XOR operation, and mem_addr[19:0] specifies the address offset into the peripheral space based at 0x2000_0000 for SRAM_U, and 0x4000_0000 for peripherals. The "-" indicates an address bit "don't care".

The decorated XOR write operation is defined in the following pseudo-code as:

```
ioxor<sz>(accessAddress, wdata)                    // decorated store XOR

tmp    =  mem[accessAddress & 0xE00FFFFF, size]   // memory read
tmp    =  tmp ^ wdata                              // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp        // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 17-3. Cycle definitions of decorated store: logical XOR

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form (rdata ^ wdata) and capture destination data in register	Perform write sending registered data to memory

17.3.1.4 Decorated store bit field insert (BFI)

This command inserts a bit field contained in the write data operand, defined by LSB position (b) and the bit field width (w+1), into the memory "container" defined by the access size associated with the store instruction using an atomic read-modify-write sequence.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

NOTE

For the word sized operation, the maximum bit field width is 16 bits. The core performs the required write data lane replication on byte and halfword transfers.

The BFI operation can be used to insert a single bit into a peripheral. For this case, the w field is simply set to 0, indicating a bit field width of 1.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2	1	0
iobfib	0	*	*	1	-	-	b	b	b	-	w	w	w	mem_addr		
iobfih	0	*	*	1	-	b	b	b	b	w	w	w	w	mem_addr		0
iobfiw	0	*	*	1	b	b	b	b	b	w	w	w	w	mem_addr	0	0

Figure 17-6. Decorated address store: bit field insert

where addr[30:29] = 01 for SRAM_U, addr[30:29] =10 for peripheral,addr[28] = 1 signals a BFI operation, addr[27:23] is "b", the LSB identifier, addr[22:19] is "w", the bit field width minus 1 identifier, and addr[18:0] specifies the address offset into the peripheral space based at 0x2000_0000 for SRAM_U, and 0x4000_0000 for peripherals.

The "-" indicates an address bit "don't care". Note, unlike the other decorated store operations, BFI uses addr[19] as the least significant bit in the "w" specifier and not as an address bit.

The decorated BFI write operation is defined in the following pseudo-code as:

```
iobfi<sz>(accessAddress, wdata)                  // decorated bit field insert

tmp   =  mem[accessAddress & 0xE007FFFF, size]   // memory read
mask  = ((1 << (w+1)) - 1) << b                  // generate bit mask
tmp   =  tmp   & ~mask                           // modify
      |  wdata &  mask
mem[accessAddress & 0xE007FFFF, size] = tmp      // memory write
```

The write data operand (wdata) associated with the store instruction contains the bit field to be inserted. It must be properly aligned within a right-aligned container, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword, or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field "xyz" into an 8-bit memory container, initially set to "abcd_efgh". For all cases, w is 2, signaling a bit field width of 3.

```
if b = 0 and the decorated store (strb) Rt register[7:0] = ----_-xyz,
   then destination is "abcd_exyz"
if b = 1 and the decorated store (strb) Rt register[7:0] = ----_xyz-,
   then destination is "abcd_xyzh"
if b = 2 and the decorated store (strb) Rt register[7:0] = ---x_yz--,
   then destination is "abcx_yzgh"
if b = 3 and the decorated store (strb) Rt register[7:0] = --xy_z---,
   then destination is "abxy_zfgh"
if b = 4 and the decorated store (strb) Rt register[7:0] = -xyz_----,
   then destination is "axyz_efgh"
if b = 5 and the decorated store (strb) Rt register[7:0] = xyz-_----,
   then destination is "xyzd_efgh"
if b = 6 and the decorated store (strb) Rt register[7:0] = yz--_----,
   then destination is "yzcd_efgh"
if b = 7 and the decorated store (strb) Rt register[7:0] = z---_----,
   then destination is "zbcd_efgh"
```

Note from the example, when the starting bit position plus the field width exceeds the container size, only part of the source bit field is inserted into the destination memory location. Stated differently, if (b + w+1) > container_width, only the low-order "container_width - b" bits are actually inserted.

The cycle-by-cycle BME operations are detailed in the following table.

Table 17-4. Cycle definitions of decorated store: bit field insert

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Convert master_wt to slave_rd; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>

Table continues on the next page...

Table 17-4. Cycle definitions of decorated store: bit field insert (continued)

Pipeline stage	Cycle		
	x	x+1	x+2
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Form bitwise ((mask) ? wdata : rdata)) and capture destination data in register	Perform write sending registered data to memory

17.3.2 BME decorated loads

The functions supported by the BME's decorated loads include two single-bit load-and-{set, clear} operators plus unsigned bit field extracts.

For the two load-and-{set, clear} operations, BME converts a single decorated AHB load transaction into a two-cycle atomic read-modify-write sequence, where the combined read-modify operations are performed in the first AHB data phase, and then the write is performed in the second AHB data phase as the original read data is returned to the processor core. For an unsigned bit field extract, the decorated load transaction is stalled for one cycle in the BME as the data field is extracted, then aligned and returned to the processor in the second AHB data phase. This is the only decorated transaction that is not an atomic read-modify-write, as it is a simple data read.

A generic timing diagram of a decorated load showing a peripheral load-and-set 1-bit operation is shown as follows.

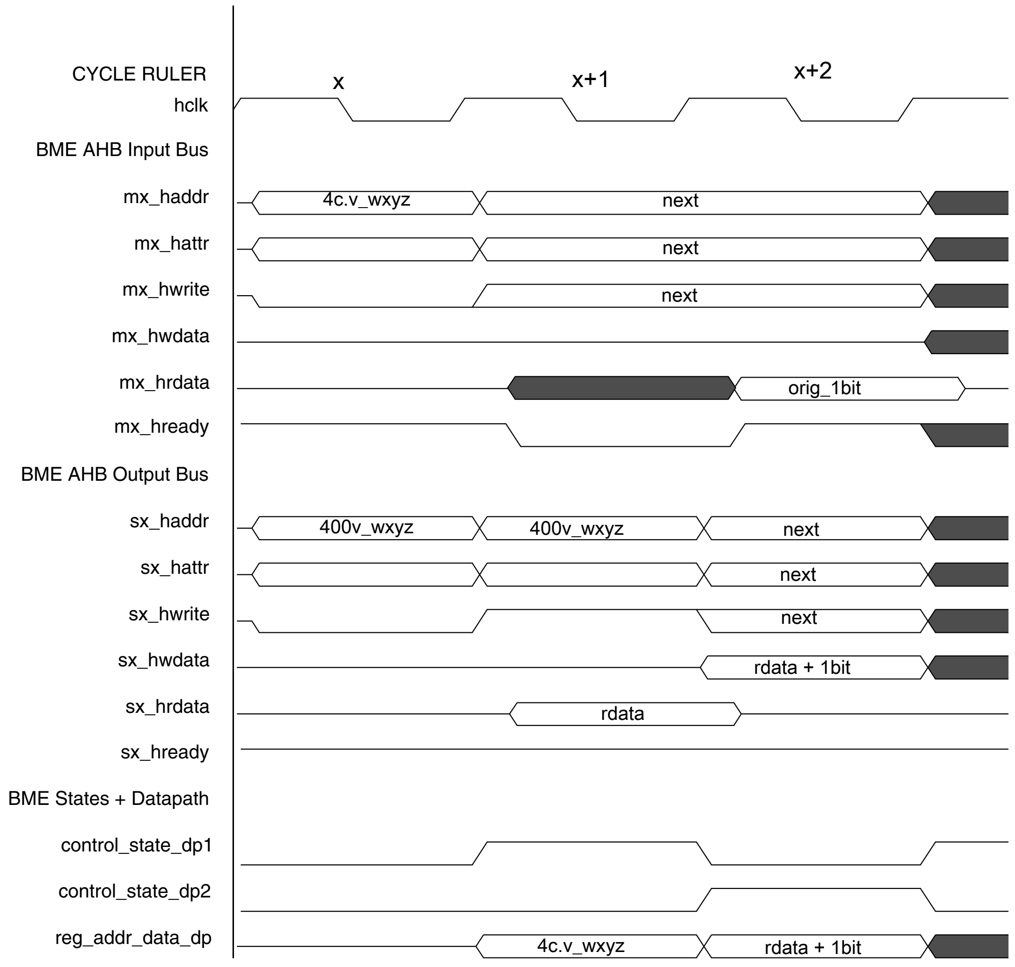


Figure 17-7. Decorated load: load-and-set 1-bit field insert timing diagram

Decorated load-and-{set, clear} 1-bit operations follow the execution template shown in the above figure: a 2-cycle read-modify-write operation:

1. Cycle x, first AHB address phase: Read from input bus is translated into a read operation on the output bus with the actual memory address (with the decoration removed) and then captured in a register
2. Cycle x+1, second AHB address phase: Write access with the registered (but actual) memory address is output
3. Cycle x+1, first AHB data phase: The "original" 1-bit memory read data is captured in a register, while the 1-bit field is set or clear based on the function defined by the decoration with the modified data captured in a register; the input bus cycle is stalled
4. Cycle x+2, second AHB data phase: The selected original 1-bit is right-justified, zero-filled and then driven onto the input read data bus, while the registered write data is sourced onto the output write data bus

NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

A generic timing diagram of a decorated load showing an unsigned peripheral bit field operation is shown in the following figure.

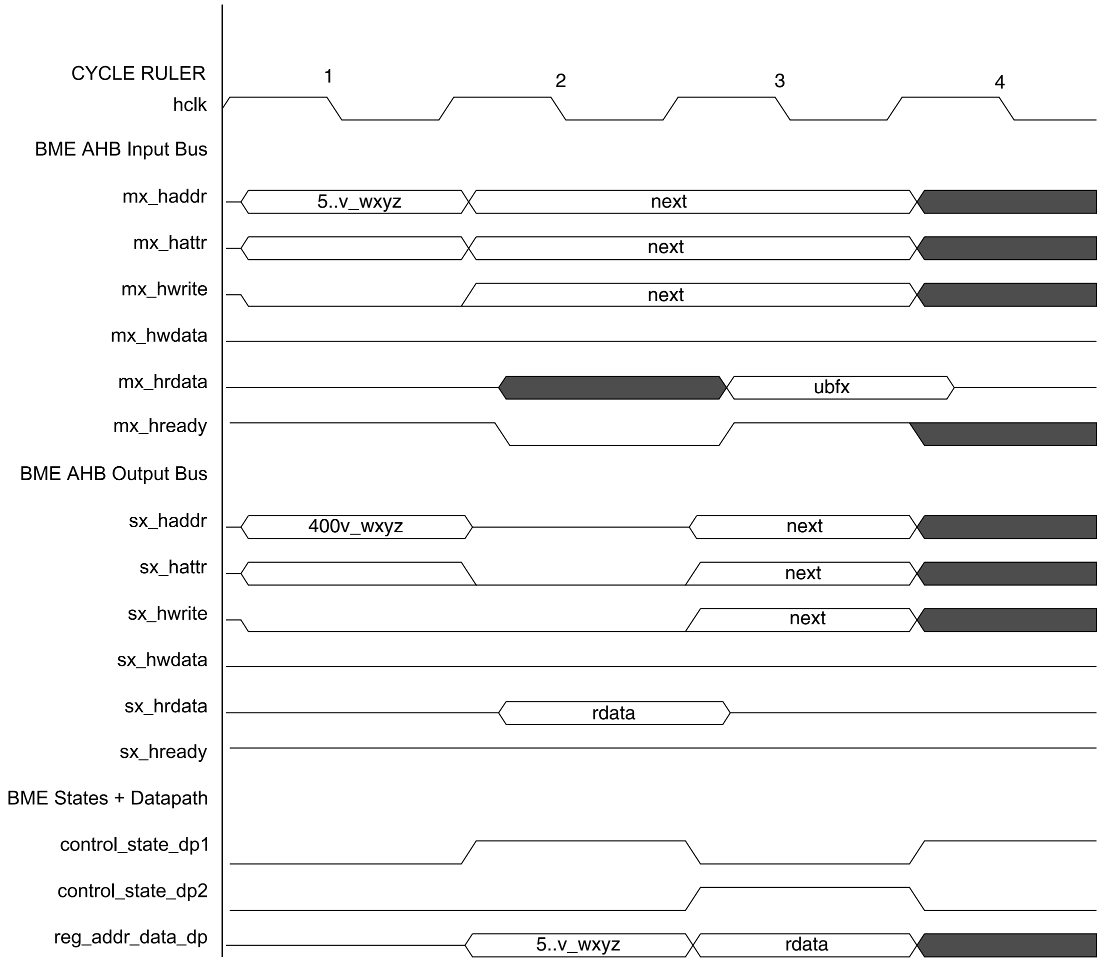


Figure 17-8. Decorated load: unsigned bit field insert timing diagram

The decorated unsigned bit field extract follows the same execution template shown in the above figure, a 2-cycle read operation:

- Cycle x, 1st AHB address phase: Read from input bus is translated into a read operation on the output bus with the actual memory address (with the decoration removed) and then captured in a register
- Cycle x+1, 2nd AHB address phase: Idle cycle

- Cycle x+1, 1st AHB data phase: A bit mask is generated based on the starting bit position and the field width; the mask is AND'ed with the memory read data to isolate the bit field; the resulting data is captured in a data register; the input bus cycle is stalled
- Cycle x+2, 2nd AHB data phase: Registered data is logically right-aligned for proper alignment and driven onto the input read data bus

NOTE

Any wait states inserted by the slave device are simply passed through the BME back to the master input bus, stalling the AHB transaction cycle for cycle.

17.3.2.1 Decorated load: load-and-clear 1 bit (LAC1)

This command loads a 1-bit field defined by the LSB position (b) into the core's general purpose destination register (Rt) and zeroes the bit in the memory space after performing an atomic read-modify-write sequence.

The extracted 1-bit data field from the memory address is right-justified and zero-filled in the operand returned to the core.

The data size is specified by the read operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

	31	30	29	28	27	26	25	24	23	22	21	20	19–2	1	0
iolac1b	0	*	*	0	1	0	-	-	b	b	b	-	mem_addr		
iolac1h	0	*	*	0	1	0	-	b	b	b	b	-	mem_addr		0
iolac1w	0	*	*	0	1	0	b	b	b	b	b	-	mem_addr	0	0

Figure 17-9. Decorated load address: load-and-clear 1 bit

See Figure 17-9 where addr[30:29] = 01 for SRAM_U, addr[30:29] = 10 for peripheral, addr[28:26] = 010 specifies the load-and-clear 1 bit operation, addr[25:21] is "b", the bit identifier, and mem_addr[19:0] specifies the address offset into the space based at 0x2000_0000 for SRAM_U, and 0x4000_0000 for peripheral. The "-" indicates an address bit "don't care".

The decorated load-and-clear 1-bit read operation is defined in the following pseudo-code as:

```
rdata =  iolac1<sz>(accessAddress)                // decorated load-and-clear 1

tmp   =  mem[accessAddress & 0xE00FFFFF, size]    // memory read
mask  =  1 << b                                   // generate bit mask
rdata = (tmp &  mask) >> b                        // read data returned to core
```

```
tmp    =  tmp & ~mask                               // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp         // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 17-5. Cycle definitions of decorated load: load-and-clear 1 bit

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Extract bit from rdata; Form (rdata & ~mask) and capture destination data in register	Return extracted bit to master; Perform write sending registered data to memory

17.3.2.2 Decorated Load: Load-and-Set 1 Bit (LAS1)

This command loads a 1-bit field defined by the LSB position (b) into the core's general purpose destination register (Rt) and sets the bit in the memory space after performing an atomic read-modify-write sequence.

The extracted one bit data field from the memory address is right justified and zero filled in the operand returned to the core.

The data size is specified by the read operation and can be byte (8-bit), halfword (16-bit) or word (32-bit).

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

Figure 17-10. Decorated load address: load-and-set 1 bit

where addr[30:29] = 01 for SRAM_U, addr[30:29] = 10 for peripheral, addr[28:26] = 011 specifies the load-and-set 1 bit operation, addr[25:21] is "b", the bit identifier, and mem_addr[19:0] specifies the address offset into the space based at 0x2000_0000 for SRAM_U, and 0x4000_0000 for peripheral. The "-" indicates an address bit "don't care".

The decorated Load-and-Set 1 Bit read operation is defined in the following pseudo-code as:

```
rdata =  iolas1<sz>(accessAddress)                  // decorated load-and-set 1
```

```
tmp    =  mem[accessAddress & 0xE00FFFFF, size]   // memory read
mask   =  1 << b                                  // generate bit mask
rdata = (tmp &  mask) >> b                        // read data returned to core
tmp    =  tmp |  mask                             // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp       // memory write
```

The cycle-by-cycle BME operations are detailed in the following table.

Table 17-6. Cycle definitions of decorated load: load-and-set 1-bit

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Recirculate captured addr + attr to memory as slave_wt	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Extract bit from rdata; Form (rdata ǀ mask) and capture destination data in register	Return extracted bit to master; Perform write sending registered data to memory

17.3.2.3 Decorated load unsigned bit field extract (UBFX)

This command extracts a bit field defined by LSB position (b) and the bit field width (w +1) from the memory "container" defined by the access size associated with the load instruction using a two-cycle read sequence.

The extracted bit field from the memory address is right-justified and zero-filled in the operand returned to the core. Recall this is the only decorated operation that does not perform a memory write, that is, UBFX only performs a read.

The data size is specified by the write operation and can be byte (8-bit), halfword (16-bit) or word (32-bit). Note for the word sized operation, the maximum bit field width is 16 bits.

The use of a UBFX operation is recommended to extract a single bit. For this case, the w field is simply set to 0, indicating a bit field width of 1.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ioubfxb	0	*	*	1	-	-	b	b	b	-	w	w	w	mem_addr																		
ioubfxh	0	*	*	1	-	b	b	b	b	w	w	w	w	mem_addr																		0
ioubfxw	0	*	*	1	b	b	b	b	b	w	w	w	w	mem_addr																	0	0

Figure 17-11. Decorated load address: unsigned bit field extract

See Figure 17-11, where addr[30:29] = 01 for SRAM_U, addr[30:29] = 10 for peripheral, addr[28] = 1 specifies the unsigned bit field extract operation, addr[27:23] is "b", the LSB identifier, addr[22:19] is "w", the bit field width minus 1 identifier, and mem_addr[18:0] specifies the address offset into the space based at 0x2000_0000 for SRAM_U, and 0x4000_0000 for peripheral. The "-" indicates an address bit "don't care". Note, unlike the other decorated load operations, UBFX uses addr[19] as the least significant bit in the "w" specifier and not as an address bit.

The decorated unsigned bit field extract read operation is defined in the following pseudo-code as:

```
rdata =  ioubfx<sz>(accessAddress)              // unsigned bit field extract

tmp   =  mem[accessAddress & 0xE007FFFF, size]  // memory read
mask  = ((1 << (w+1)) - 1) << b                 // generate bit mask
rdata = (tmp &  mask) >> b                      // read data returned to core
```

Like the BFI operation, when the starting bit position plus the field width exceeds the container size, only part of the source bit field is extracted from the destination memory location. Stated differently, if (b + w+1) > container_width, only the low-order "container_width - b" bits are actually extracted. The cycle-by-cycle BME operations are detailed in the following table.

Table 17-7. Cycle definitions of decorated load: unsigned bit field extract

Pipeline Stage	Cycle		
	x	x+1	x+2
BME AHB_ap	Forward addr to memory; Decode decoration; Capture address, attributes	Idle AHB address phase	<next>
BME AHB_dp	<previous>	Perform memory read; Form bit mask; Form (rdata & mask) and capture destination data in register	Logically right shift registered data; Return justified rdata to master

17.3.3 Additional details on decorated addresses and GPIO accesses

As previously noted, the peripheral address space occupies a 516 KB region: 512 KB based at 0x4000_0000 plus a 4 KB space based at 0x400F_F000 for GPIO accesses. This memory layout provides compatibility with the Kinetis K Family and provides 129 address "slots", each 4 KB in size.

The GPIO address space is multiply-mapped by the hardware: it appears at the "standard" system address 0x400F_F000 and is physically located in the address slot corresponding to address 0x4000_F000. Decorated loads and stores create a slight complication involving accesses to the GPIO. Recall the use of address[19] varies by decorated operation; for AND, OR, XOR, LAC1 and LAS1, this bit functions as a true address bit, while for BFI and UBFX, this bit defines the least significant bit of the "w" bit field specifier.

As a result, undecorated GPIO references and decorated AND, OR, XOR, LAC1 and LAS1 operations can use the standard 0x400F_F000 base address, while decorated BFI and UBFX operations must use the alternate 0x4000_F000 base address. Another implementation can simply use 0x400F_F000 as the base address for all undecorated GPIO accesses and 0x4000_F000 as the base address for all decorated accesses. Both implementations are supported by the hardware.

Table 17-8. Decorated peripheral and GPIO address details

Peripheral address space	Description
0x4000_0000–0x4007_FFFF	Undecorated (normal) peripheral accesses
0x4008_0000–0x400F_EFFF	Illegal addresses; attempted references are aborted and error terminated
0x400F_F000–0x400F_FFFF	Undecorated (normal) GPIO accesses using standard address
0x4010_0000–0x43FF_FFFF	Illegal addresses; attempted references are aborted and error terminated
0x4400_0000–0x4FFF_FFFF	Decorated AND, OR, XOR, LAC1, LAS1 references to peripherals and GPIO based at either 0x4000_F000 or 0x400F_F000
0x5000_0000–0x5FFF_FFFF	Decorated BFI, UBFX references to peripherals and GPIO only based at 0x4000_F000

17.4 Application information

In this section, GNU assembler macros with C expression operands are presented as examples of the required instructions to perform decorated operations.

This section specifically presents a partial bme.h file defining the assembly language expressions for decorated logical stores: AND, OR, and XOR. Comparable functions for BFI and the decorated loads are more complex and available in the complete BME header file.

These macros use the same function names presented in Functional description.

```
#define IOANDW(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "str    r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOANDH(ADDR,WDATA)          \
    __asm("ldr    r3, =(1<<26);"    \
          "orr    r3, %[addr];"     \
          "mov    r2, %[wdata];"    \
          "strh   r2, [r3];"        \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
```

```
#define IOANDB(ADDR,WDATA)          \
    __asm("ldr     r3, =(1<<26);"   \
          "orr     r3, %[addr];"    \
          "mov     r2, %[wdata];"   \
          "strb    r2, [r3];"       \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOORW(ADDR,WDATA)           \
    __asm("ldr     r3, =(1<<27);"   \
          "orr     r3, %[addr];"    \
          "mov     r2, %[wdata];"   \
          "str     r2, [r3];"       \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOORH(ADDR,WDATA)           \
    __asm("ldr     r3, =(1<<27);"   \
          "orr     r3, %[addr];"    \
          "mov     r2, %[wdata];"   \
          "strh    r2, [r3];"       \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOORB(ADDR,WDATA)           \
    __asm("ldr     r3, =(1<<27);"   \
          "orr     r3, %[addr];"    \
          "mov     r2, %[wdata];"   \
          "strb    r2, [r3];"       \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORW(ADDR,WDATA)          \
    __asm("ldr     r3, =(3<<26);"   \
          "orr     r3, %[addr];"    \
          "mov     r2, %[wdata];"   \
          "str     r2, [r3];"       \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORH(ADDR,WDATA)          \
    __asm("ldr     r3, =(3<<26);"   \
          "orr     r3, %[addr];"    \
          "mov     r2, %[wdata];"   \
          "strh    r2, [r3];"       \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");

#define IOXORB(ADDR,WDATA)          \
    __asm("ldr     r3, =(3<<26);"   \
          "orr     r3, %[addr];"    \
          "mov     r2, %[wdata];"   \
          "strb    r2, [r3];"       \
          :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
```

Chapter 18
Flash Memory Module (FTMRE)

18.1 Introduction

The FTMRE module implements the following:

- Program flash (flash) memory

The flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The flash module includes a memory controller that executes commands to modify flash memory contents. The user interface to the memory controller consists of the indexed Flash Common Command Object (FCCOB) register which is written to with the command, global address, data, and any required command parameters. The memory controller must complete the execution of a command before the FCCOB register can be written to with a new command.

CAUTION

A flash byte or longword must be in the erased state before being programmed. Cumulative programming of bits within a flash byte or longword is not allowed.

The flash memory is read as bytes. Read access time is one bus cycle for bytes. For flash memory, an erased bit reads 1 and a programmed bit reads 0.

18.2 Feature

18.2.1 Flash memory features

The flash memory has the following features:

- 8 KB of flash memory composed of one 8 KB flash block divided into 16 sectors of 512 bytes
- Automated program and erase algorithm with verify

- Fast sector erase and longword program operation
- Flexible protection scheme to prevent accidental programming or erasing of flash memory

18.2.2 Other flash module features

The flash memory module has the following other features:

- No external high-voltage power supply required for flash memory program and erase operations
- Interrupt generation on flash command completion
- Security mechanism to prevent unauthorized access to the flash memory

18.3 Functional description

18.3.1 Modes of operation

The flash memory module provides the normal user mode of operation. The operating mode is determined by module-level inputs and affects the FCNFG, and FCLKDIV registers.

18.3.1.1 Wait mode

The flash memory module is not affected if the MCU enters Wait mode. The flash module can recover the MCU from Wait via the CCIF interrupt. See Flash interrupts.

18.3.1.2 Stop mode

If a flash command is active, that is, FSTAT[CCIF] = 0, when the MCU requests Stop mode, the current NVM operation will be completed before the MCU is allowed to enter Stop mode.

18.3.2 Flash memory map

The MCU places the flash memory as shown in the following table.

MKE04Z8 contains a piece of 8 KB flash memory. This flash block is divided into 16 sectors of 512 bytes.

Table 18-1. Flash memory addressing

Device	Global address	Size (Bytes)	Description

MKE04Z8VTG4(R)
MKE04Z8VWJ4(R)
MKE04Z8VFK4(R)
	0x0000–0x1FFF	8 KB	Flash block contains flash configuration field.

18.3.3 Flash initialization after system reset

On each system reset, the flash module executes an initialization sequence that establishes initial values for the flash block configuration parameters, the FPROT protection register, and the FOPT and FSEC registers. The initialization routine reverts to built-in default values that leave the module in a fully protected and secured state if errors are encountered during execution of the reset sequence. If an error is detected during the reset sequence, both FSTAT[MGSTAT] bits will be set.

FSTAT[CCIF] is cleared throughout the initialization sequence. The NVM module holds off all CPU access for a portion of the initialization sequence. Flash reads are allowed after the hold is removed. Completion of the initialization sequence is marked by setting FSTAT[CCIF] high, which enables user commands. While FSTAT[CCIF] remains cleared, it is not possible to write on registers FCCOBIX or FCCOB.

If a reset occurs while any flash command is in progress, that command will be immediately aborted. The state of the word being programmed or the sector/block being erased is not guaranteed.

18.3.4 Flash command operations

Flash command operations are used to modify flash memory contents.

The command operations contain three steps:

1. Configure the clock for flash program and erase command operations.
2. Use command write sequence to set flash command parameters and launch execution.
3. Execute valid flash commands according to MCU functional mode and MCU security state.

The figure below shows a general flowchart of the flash command write sequence.

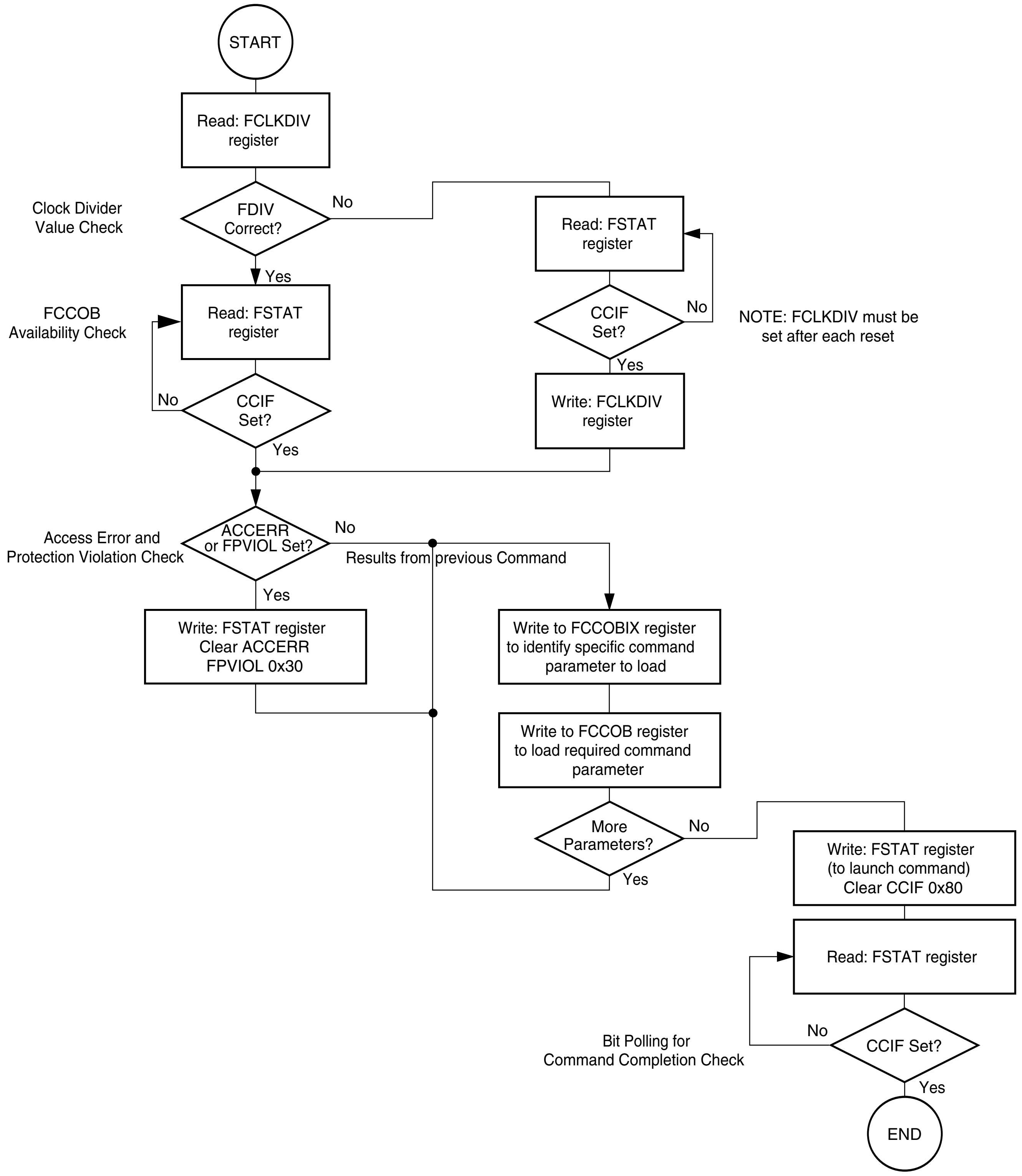

START
Read: FCLKDIV register
Clock Divider Value Check
FDIV Correct?
No
Yes
Read: FSTAT register
CCIF Set?
No
Yes
Write: FCLKDIV register
NOTE: FCLKDIV must be set after each reset
FCCOB Availability Check
Read: FSTAT register
CCIF Set?
No
Yes
Access Error and Protection Violation Check
ACCERR or FPVIOL Set?
No
Yes
Results from previous Command
Write: FSTAT register Clear ACCERR FPVIOL 0x30
Write to FCCOBIX register to identify specific command parameter to load
Write to FCCOB register to load required command parameter
More Parameters?
No
Yes
Write: FSTAT register (to launch command) Clear CCIF 0x80
Read: FSTAT register
Bit Polling for Command Completion Check
CCIF Set?
No
Yes
END

Figure 18-1. Generic flash command write sequence flowchart

18.3.4.1 Writing the FCLKDIV register

Prior to issuing any flash program or erase command after a reset, the user is required to write the FCLKDIV register to divide BUSCLK down to a target FCLK of 1 MHz. The following table shows recommended values for FCLKDIV[FDIV] based on BUSCLK frequency.

Table 18-2. FDIV values for various BUSCLK frequencies

BUSCLK frequency (MHz)		FDIV[5:0]
MIN[1]	MAX[2]	
1.0	1.6	0x00
1.6	2.6	0x01
2.6	3.6	0x02
3.6	4.6	0x03
4.6	5.6	0x04
5.6	6.6	0x05
6.6	7.6	0x06
7.6	8.6	0x07
8.6	9.6	0x08
9.6	10.6	0x09
10.6	11.6	0x0A
11.6	12.6	0x0B
12.6	13.6	0x0C
13.6	14.6	0x0D
14.6	15.6	0x0E
15.6	16.6	0x0F
16.6	17.6	0x10
17.6	18.6	0x11
18.6	19.6	0x12
19.6	20.6	0x13
20.6	21.6	0x14
21.6	22.6	0x15
22.6	23.6	0x16
23.6	24.6	0x17
24.6	25.6	0x18

1. BUSCLK is greater than this value.
2. BUSCLK is less than or equal to this value.

CAUTION

Programming or erasing the flash memory cannot be performed if the bus clock runs at less than 0.8 MHz. Setting FCLKDIV[FDIV] too high can destroy the flash memory due to overstress. Setting FCLKDIV[FDIV] too low can result in incomplete programming or erasure of the flash memory cells.

When the FCLKDIV register is written, FCLKDIV[FDIVLD] is set automatically. If FCLKDIV[FDIVLD] is 0, the FCLKDIV register has not been written since the last reset. If the FCLKDIV register has not been written, any flash program or erase command loaded during a command write sequence will not execute and FSTAT[ACCERR] will be set.

18.3.4.2 Command write sequence

The memory controller will launch all valid flash commands entered using a command write sequence.

Before launching a command, FSTAT[ACCERR] and FSTAT[FPVIOL] must be cleared and the FSTAT[CCIF] flag will be tested to determine the status of the current command write sequence. If FSTAT[CCIF] is 0, indicating that the previous command write sequence is still active, a new command write sequence cannot be started and all writes to the FCCOB register are ignored.

The FCCOB parameter fields must be loaded with all required parameters for the flash command being executed. Access to the FCCOB parameter fields is controlled via FCCOBIX[CCOBIX].

Flash command mode uses the indexed FCCOB register to provide a command code and its relevant parameters to the memory controller. First, the user must set up all required FCCOB fields. Then they can initiate the command's execution by writing a 1 to FSTAT[CCIF]. This action clears the CCIF command completion flag to 0. When the user clears FSTAT[CCIF], all FCCOB parameter fields are locked and cannot be changed by the user until the command completes (evidenced by the memory controller returning FSTAT[CCIF] to1). Some commands return information to the FCCOB register array.

The generic format for the FCCOB parameter fields in flash command mode is shown in the following table. The return values are available for reading after the FSTAT[CCIF] flag has been returned to 1 by the memory controller. Writes to the unimplemented parameter fields, FCCOBIX[CCOBIX] =110b and FCCOBIX[CCOBIX] = 111b, are ignored with read from these fields returning 0x0000.

Table 18-3 shows the generic flash command format. The high byte of the first word in the CCOB array contains the command code, followed by the parameters for this specific flash command. For details on the FCCOB settings required by each command, see the flash command descriptions in Flash command summary .

Table 18-3. FCCOB – flash command mode typical usage

CCOBIX[2:0]	Byte	FCCOB parameter fields in flash command mode
000	HI	FCMD[7:0] defining flash command
	LO	Global address [23:16]
001	HI	Global address [15:8]
	LO	Global address [7:0]
010	HI	Data 0 [15:8]
	LO	Data 0 [7:0]
011	HI	Data 1 [15:8]
	LO	Data 1 [7:0]
100	HI	Data 2 [15:8]
	LO	Data 2 [7:0]
101	HI	Data 3 [15:8]
	LO	Data 3 [7:0]

The contents of the FCCOB parameter fields are transferred to the memory controller when the user clears the FSTAT[CCIF] command completion flag by writing 1. The CCIF flag will remain clear until the flash command has completed. Upon completion, the memory controller will return FSTAT[CCIF] to 1 and the FCCOB register will be used to communicate any results.

The following table presents the valid flash commands, as enabled by the combination of the functional MCU mode with the MCU security state of unsecured or secured.

MCU secured state is selected by FSEC[SEC].

Table 18-4. Flash commands by mode and security state

FCMD	Command	Unsecured	Secured
		U[1]	U[2]
0x01	Erase verify all blocks	*	*
0x02	Erase verify block	*	*
0x03	Erase verify flash section	*	*
0x04	Read once	*	*
0x06	Program flash	*	*
0x07	Program once	*	*
0x08	Erase all block	*	*
0x09	Erase flash block	*	*

Table continues on the next page...

Table 18-4. Flash commands by mode and security state (continued)

FCMD	Command	Unsecured	Secured
		U[1]	U[2]
0x0A	Erase flash sector	*	*
0x0B	Unsecure flash	*	*
0x0C	Verify backdoor access key	*	*
0x0D	Set user margin level	*	*
0x0E	Set factory margin level	*	*

1. Unsecured user mode
2. Secured user mode

18.3.5 Flash interrupts

The flash module can generate an interrupt when a flash command operation has completed.

Table 18-5. Flash interrupt source

Interrupt source	Interrupt flag	Local enable	Global (CCR) mask
Flash command complete	CCIF (FSTAT register)	CCIE (FCNFG register)	I Bit

18.3.5.1 Description of flash interrupt operation

The flash module uses the FSTAT[CCIF] flag in combination with the FCNFG[CCIE] interrupt enable bit to generate the flash command interrupt request.

The logic used for generating the flash module interrupts is shown in the following figure.

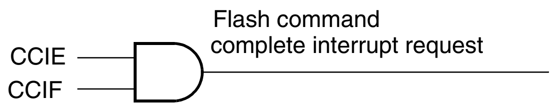


Figure 18-2. Flash module interrupts implementation

18.3.6 Protection

The FPROT register can be set to protect regions in the flash memory from accidental programing or erasing. The memory region growing upward from global address 0x0000 in the flash memory can be activated for protection. The flash memory addresses covered by these protectable regions are shown in the flash memory map.

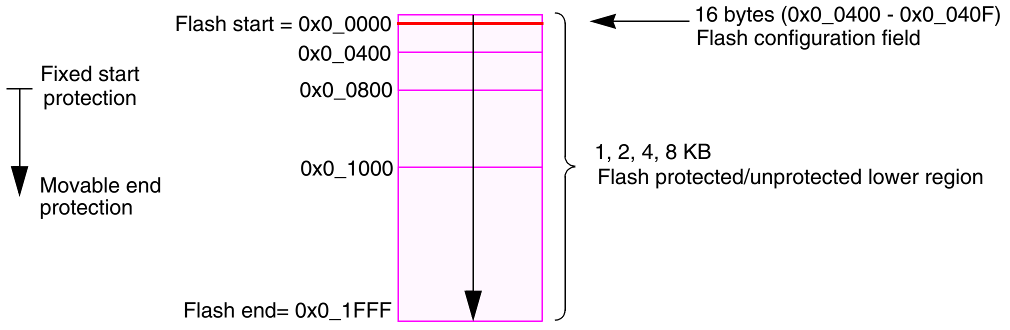


Figure 18-3. 8 KB flash protection memory map

Default protection settings as well as security information that allows the MCU to restrict access to the flash module are stored in the flash configuration field as described in the table below.

Table 18-6. Flash configuration field

Global address	Size (Bytes)	Description
0x0400–0x0407	8	Backdoor comparison key. See Verify backdoor access key command and Unsecuring the MCU using backdoor key access.
0x0408–0x040B [1]	4	Reserved
0x040C–0x040F[1]	1	Flash nonvolatile byte - data[31:24]
	1	Flash security byte - data[23:16]
	1	Flash protection byte - data[15:8]
	1	Reserved - data [7:0]

1. 0x0_0408–0x040B and 0x040C–0x0_040F form a flash longword in each address range and must be programmed in a single command write sequence. Each byte in these longwords that are marked as reserved must be programmed to 0xFF. Alternatively, the Flash phrase 0x0408-0x040F can also be programmed in a single command write sequence.

The flash module provides protection to the MCU. During the reset sequence, the FPROT register is loaded with the contents of the flash protection byte in the flash configuration field at global address 0x040D in flash memory. The protection functions depend on the configuration of bit settings in FPROT register.

Table 18-7. Flash protection function

FPOPEN	FPLDIS	Function
1	1	No flash protection
1	0	Protected low range
0	1	Full flash memory protected
0	0	Unprotected low range

The flash protection scheme can be used by applications requiring reprogramming in single chip mode while providing as much protection as possible if reprogramming is not required.

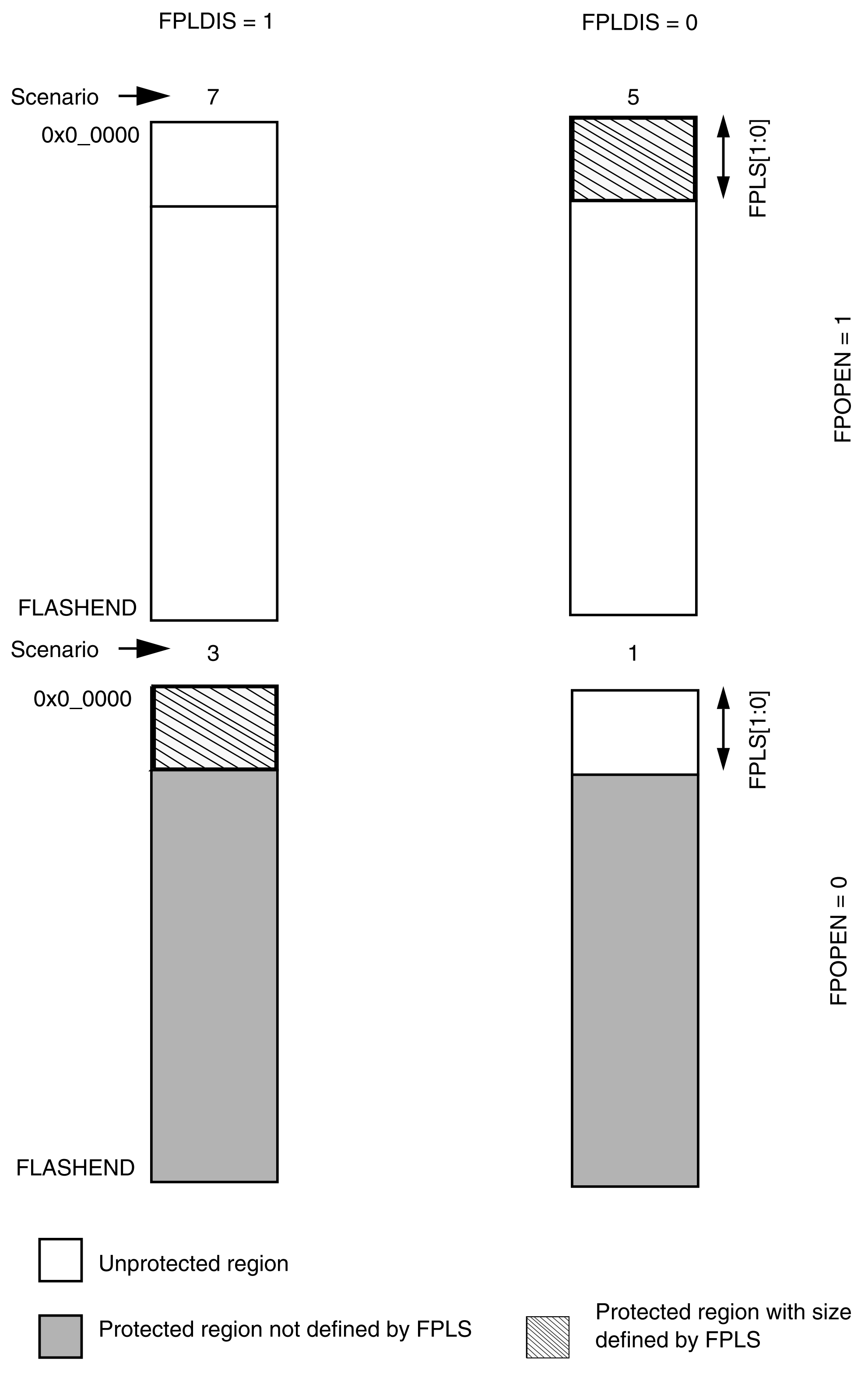


Figure 18-4. Flash protection scenarios

The general guideline is that flash protection can only be added and not removed. The following table specifies all valid transitions between flash protection scenarios. Any attempt to write an invalid scenario to the FPROT register will be ignored. The contents of the FPROT register reflect the active protection scenario. See the FPROT[FPLS] field descriptions for additional restrictions.

Table 18-8. Flash protection scenario transitions

From protection scenario	To protection scenario[1]			
	1	3	5	7
1	×	×		
3		×		
5		×	×	
7	×	×	×	×

1. Allowed transitions marked with X.

The flash protection address range is listed in the following two tables regarding the scenarios in the table above.

Table 18-9. Flash protection lower address range

FPLS[1:0]	Global address range	Protected size
00	0x0000–0x03FF	1 KB
01	0x0000–0x07FF	2 KB
10	0x0000–0x0FFF	4 KB
11	0x0000–0x1FFF	8 KB

All possible flash protection scenarios are shown in Figure 18-4. Although the protection scheme is loaded from the flash memory at global address 0x040D during the reset sequence, it can be changed by the user.

18.3.7 Security

The flash module provides security information to the MCU. The flash security state is defined by FSEC[SEC]. During reset, the flash module initializes the FSEC register using data read from the security byte of the flash configuration field. The security state out of reset can be permanently changed by programming the security byte, assuming that the MCU is starting from a mode where the necessary flash erase and program commands are available and that the upper region of the flash is unprotected. If the flash security byte is successfully programmed, its new value will take effect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using backdoor key access
- Unsecuring the MCU using SWD
- Mode and security effects on flash command availability

18.3.7.1 Unsecuring the MCU using backdoor key access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys, which are four 16-bit words programmed at addresses 0x400–0x407. If the KEYEN[1:0] bits are in the enabled state, the verify backdoor access key command – see Verify backdoor access key command, allows the user to present four prospective keys for comparison to the keys stored in the flash memory via the memory controller. If the keys presented in the verify backdoor access key command match the backdoor keys stored in the flash memory, FSEC[SEC] will be changed to unsecure the MCU. Key values of 0x0000 and 0xFFFF are not permitted as backdoor keys. While the Verify Backdoor Access Key command is active, flash memory will not be available for read access and will return invalid data.

The user code stored in the flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state, the MCU can be unsecured by the backdoor key access sequence described below:

1. Follow the command sequence for the verify backdoor access key command as explained in Verify backdoor access key command.
2. If the verify backdoor access key command is successful, the MCU is unsecured and FSEC[SEC] is forced to the unsecure state of 10.

The verify backdoor access key command is monitored by the memory controller and an illegal key will prohibit future use of the verify backdoor access key command. A reset of the MCU is the only method to re-enable the verify backdoor access key command. The security as defined in the flash security byte is not changed by using the verify backdoor access key command sequence. The backdoor keys stored in addresses 0x400–0x407 are unaffected by the verify backdoor access key command sequence. The verify backdoor access key command sequence has no effect on the program and erase protections defined in the flash protection register, FPROT.

After the backdoor keys have been correctly matched, the MCU will be unsecured. After the MCU is unsecured, the sector containing the flash security byte can be erased and the flash security byte can be reprogrammed to the unsecure state, if desired. In the unsecure state, the user has full control of the contents of the backdoor keys by programming addresses 0x400–0x407 in the flash configuration field.

18.3.7.2 Unsecuring the MCU using SWD

A secured MCU can be unsecured by using the following method to erase the flash memory:

1. Reset the device by asserting $\overline{\text{RESET}}$ pin or DAP_CTRL[3].
2. Set DAP_CTRL[0] bit to invoke debug mass erase via SWD
3. Release reset by deasserting $\overline{\text{RESET}}$ pin or DAP_CTRL[3] bit via SWD.
4. Wait till DAP_CTRL[0] bit is cleared (After mass erase completes, DAP_CTRL[0] bit is cleared automatically). At this time, CPU will be in hold state, MASS erase is completed, and the device is in unsecure state (flash security byte in flash configuration field is programmed with 0xFE) .
5. Reset the device.

18.3.7.3 Mode and security effects on flash command availability

The availability of flash module commands depends on the MCU operating mode and security state as shown in Table 18-4.

18.3.8 Flash commands

18.3.8.1 Flash commands

The following table summarizes the valid flash commands as well as the effects of the commands on the flash block and other resources within the flash module.

Table 18-10. Flash commands

FCMD	Command	Function on flash memory
0x01	Erase Verify All Blocks	Verifies that all flash blocks are erased
0x02	Erase Verify Block	Verifies that a flash block is erased
0x03	Erase Verify Flash Section	Verifies that a given number of words starting at the address provided are erased

Table continues on the next page...

Table 18-10. Flash commands (continued)

FCMD	Command	Function on flash memory
0x04	Read Once	Reads a dedicated 64-byte field in the nonvolatile information register in flash block that was previously programmed using the program once command
0x06	Program Flash	Programs up to two longwords in a flash block
0x07	Program Once	Programs a dedicated 64 byte field in the nonvolatile information register in flash block that is allowed to be programmed only once
0x08	Erase All Block	Erases all flash blocks An erase of all flash blocks is possible only when the FPROT[FPHDIS], FPROT[FPLDIS] and FPROT[FPOEN] and the bit are set prior to launching the command
0x09	Erase Flash Block	Erases a flash block An erase of the full flash block is possible only when FPROT[FPLDIS], FPROT[FPHDIS], and FPROT[FPOEN] are set prior to launching the command.
0x0A	Erase Flash Sector	Erases all bytes in a flash sector
0x0B	Unsecure Flash	Supports a method of releasing MCU security by erasing all flash blocks and verifying that all flash blocks are erased
0x0C	Verify Backdoor Access key	Supports a method of releasing MCU security by verifying a set of security keys
0x0D	Set User Margin Level	Specifies a user margin read level for all flash blocks
0x0E	Set Factory Margin Level	Specifies a factory margin read level for all flash blocks

18.3.9 Flash command summary

This section provides details of all available flash commands launched by a command write sequence. The FSTAT[ACCERR] will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the memory controller:

- Starting any command write sequence that programs or erases flash memory before initializing the FLCKDIV register.
- Writing an invalid command as part of the command write sequence.
- For additional possible errors, refer to the error handling table provided for each command.

If a flash block is read during the execution of an algorithm (FSTAT[CCIF] = 0) on that same block, the read operation will return invalid data. It will also trigger an illegal access exception.

If FSTAT[ACCERR] or FSTAT[FPVIOL] are set, the user must clear these fields before starting any command write sequence.

CAUTION

An flash longword must be in the erased state before being programmed. Cumulative programming of bits within an flash longword is not allowed.

18.3.9.1 Erase Verify All Blocks command

The Erase Verify All Blocks command will verify that all flash blocks have been erased.

Table 18-11. Erase Verify All Blocks command FCCOB requirements

CCOBIX[2:0]	FCCOBHI parameters	FCCOBLO parameters
000	0x01	Not required

Upon clearing FSTAT[CCIF] to launch the Erase Verify All Blocks command, the memory controller will verify that the entire flash memory space is erased. The FSTAT[CCIF] flag will set after the erase verify all blocks operation has completed. If all blocks are not erased, it means blank check failed and both FSTAT[MGSTAT] bits will be set.

Table 18-12. Erase verify all blocks command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 000 at command launch
	FPVIOL	None
	MGSTAT1	Set if any errors have been encountered during the read[1] or if blank check failed
	MGSTAT0	Set if any non-correctable errors have been encountered during the read or if blank check failed

1. As found in the memory map for NVM

18.3.9.2 Erase Verify Block command

The Erase Verify Block command allows the user to verify that an entire flash block has been erased. The FCCOB global address [23:0] bits determine which block must be verified.

Table 18-13. Erase Verify Block Command FCCOB requirements

CCOBIX[2:0]	FCCOBHI parameters	FCCOBLO parameters
000	0x02	Global address [23:16] to identify flash block
001	Global address [15:0] in flash block to be verified	

Upon clearing FSTAT[CCIF] to launch the erase verify block command, the memory controller will verify that the selected flash block is erased. The FSTAT[CCIF] flag will set after the erase verify block operation has completed. If the block is not erased, it means blank check failed and both FSTAT[MGSTAT] bits will be set.

Table 18-14. Erase Verify Block command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 000 at command launch
		Set if an invalid global address [23:0] is supplied[1]
	FPVIOL	None
	MGSTAT1	Set if any errors have been encountered during the read or if blank check failed
	MGSTAT0	Set if any non-correctable errors have been encountered during the read or if blank check failed

1. As found in the memory map for NVM

18.3.9.3 Erase Verify Flash Section command

The Erase Verify Flash Section command will verify that a section of code in the flash memory is erased. The Erase Verify Flash Section command defines the starting point of the code to be verified and the number of longwords.

Table 18-15. Erase verify flash section command FCCOB requirements

CCOBIX[2:0]	FCCOBHI parameters	FCCOBLO parameters
000	0x03	Global address [23:16] of flash block
001	Global address [15:0] of the first longwords to be verified	
010	Number of long words to be verified	

Upon clearing FSTAT[CCIF] to launch the erase verify flash section command, the memory controller will verify that the selected section of flash memory is erased. The FSTAT[CCIF] flag will set after the erase verify flash section operation has completed. If the section is not erased, it means blank check failed and both FSTAT[MGSTAT] bits will be set.

Table 18-16. Erase Verify Flash Section command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 010 at command launch
		Set if command not available in current mode (see Table 18-4)
		Set if an invalid global address [23:0] is supplied (see Table 18-1)[1]

Table continues on the next page...

Table 18-16. Erase Verify Flash Section command error handling (continued)

Register	Error bit	Error condition
		Set if a misaligned long words address is supplied (global address[1:0] != 00)
		Set if the requested section crosses flash address boundary
	FPVIOL	None
	MGSTAT1	Set if any errors have been encountered during the read[2] or if blank check failed
	MGSTAT0	Set if any non-correctable errors have been encountered during the read[2] or if blank check failed

1. As defined by the memory map for NVM
2. As found in the memory map for NVM

18.3.9.4 Read once command

The read once command provides read access to a reserved 64-byte field (8 phrase) located in the nonvolatile information register of flash. The read once field can only be programmed once and can not be erased. It can be used to store the product ID or any other information that can be written only once. It is programmed using the program once command described in Program Once command. To avoid code runaway, the read once command must not be executed from the flash block containing the program once reserved field.

Table 18-17. Read Once command FCCOB requirements

CCOBIX[2:0]	FCCOB parameters	
000	0x04	Not required
001	Read once phrase index (0x0000 – 0x0007)	
010	Read once word 0 value	
011	Read once word 1 value	
100	Read once word 2 value	
101	Read once word 3 value	

Upon clearing FSTAT[CCIF] to launch the read once command, a read once phrase is fetched and stored in the FCCOB indexed register. The FSTAT[CCIF] flag will set after the read once operation has completed. Valid phrase index values for the read once command range from 0x0000 to 0x0007. During execution of the read once command, any attempt to read addresses within flash block will return invalid data.

Table 18-18. Read Once command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 001 at command launch
		Set if command is not available in current mode (see Table 18-4)
		Set if an invalid phrase index is supplied
	FPVIOL	None
	MGSTAT1	Set if any errors have been encountered during the read
	MGSTAT0	Set if any non-correctable errors have been encountered during the read

18.3.9.5 Program Flash command

The program flash operation will program up to two previously erased longwords in the flash memory using an embedded algorithm.

Note

A flash phrase must be in the erased state before being programmed. Cumulative programming of bits within a flash phrase is not allowed.

Table 18-19. Program Flash command FCCOB requirements

CCOBIX[2:0]	FCCOBHI parameters	FCCOBLO parameters
000	0x06	Global address [23:16] to identify flash block
001	Global address [15:0] of longwords location to be programmed[1]	
010	Word 0 (longword 0) program value	
011	Word 1 (longword 0) program value	
100	Word 2 (longword 1) program value	
101	Word 3 (longword 1) program value	

1. Global address [1:0] must be 00.

Upon clearing FSTAT[CCIF] to launch the Program Flash command, the memory controller will program the data words to the supplied global address and will then proceed to verify the data words read back as expected. The FSTAT[CCIF] flag will set after the program flash operation has completed.

Table 18-20. Program Flash command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] ≠ 011 or 101 at command launch
		Set if command not available in current mode (see Table 18-4)
		Set if an invalid global address [23:0] is supplied (see Table 18-1.[1]

Table continues on the next page...

Table 18-20. Program Flash command error handling (continued)

Register	Error bit	Error condition
		Set if a misaligned longword address is supplied (global address [1:0] != 00)
		Set if the requested group of words breaches the end of the flash block.
	FPVIOL	Set if the global address [23:0] points to a protected data
	MGSTAT1	Set if any errors have been encountered during the verify operation
	MGSTAT0	Set if any non-correctable errors have been encountered during the verify operation

1. As defined by the memory map of NVM.

18.3.9.6 Program Once command

The Program Once command restricts programming to a reserved 64-byte field (8 phrases) in the nonvolatile information register located in flash. The program once reserved field can be read using the read once command as described in Read once command. The program once command must be issued only once because the nonvolatile information register in flash cannot be erased. To avoid code runaway, the program once command must not be executed from the flash block containing the program once reserved field.

Table 18-21. Program Once command FCCOB requirements

CCOBIX[2:0]	FCCOB parameters	
000	0x07	Not required
001	Program Once phrase index (0x000 – 0x0007)	
010	Program once Word 0 value	
011	Program once Word 1value	
100	Program once Word 2 value	
101	Program once Word 3 value	

Upon clearing FSTAT[CCIF] to launch the program once command, the memory controller first verifies that the selected phrase is erased. If erased, then the selected phrase will be programmed and then verified with read back. The FSTAT[CCIF] flag will remain clear, setting only after the program once operation has completed.

The reserved nonvolatile information register accessed by the Program Once command cannot be erased, and any attempt to program one of these phrases a second time will not be allowed. Valid phrase index values for the program once command range from 0x0000 to 0x0007. During execution of the program once command, any attempt to read addresses within flash will return invalid data.

Table 18-22. Program Once ommand error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 101 at command launch
		Set if command not available in current mode (see Table 18-4)
		Set if an invalid phrase index is supplied
		Set if the requested phrase has already been programmed[1]
	FPVIOL	None
	MGSTAT1	Set if any errors have been encountered during the verify operation
	MGSTAT0	Set if any non-correctable errors have been encountered during the verify operation

1. If a program once phrase is initially programmed to 0xFFFF_FFFF_FFFF_FFFF, the program once command will be allowed to execute again on that same phrase.

18.3.9.7 Erase All Blocks command

The Erase All Blocks operation will erase the entire flash memory space.

Table 18-23. Erase All Blocks command FCCOB requirements

CCOBIX[2:0]	FCCOBHI parameters	FCCOBLO parameters
000	0x08	Not required

Upon clearing FSTAT[CCIF] to launch the Erase All Blocks command, the memory controller will erase the entire NVM memory space and verify that it is erased. If the memory controller verifies that the entire NVM memory space was properly erased, security will be released. Therefore, the device is in unsecured state. During the execution of this command (FSTAT[CCIF] = 0) the user must not write to any NVM module register. The FSTAT[CCIF] flag will set after the erase all blocks operation has completed.

Table 18-24. Erase All Blocks command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] ≠ 000 at command launch
		Set if command not available in current mode (see Table 18-4)
	FPVIOL	Set if any area of the flash memory is protected
	MGSTAT1	Set if any errors have been encountered during the verify operation[1]
	MGSTAT0	Set if any non-correctable errors have been encountered during the verify operation[1]

1. As found in the memory map for NVM.

18.3.9.8 Debugger mass erase request

The functionality of the Erase All Blocks command is also available in an uncommanded fashion from the Debugger Mass Erase Request feature.

The Debugger Mass Erase request requires the clock divider register FCLKDIV to be loaded before invoking this function. Please look into the Reference Manual for information about the default value of FCLKDIV in case direct writes to register FCLKDIV are not allowed by the time this feature is invoked. If FCLKDIV is not set the Debugger Mass Erase request will not execute and the FSTAT[ACCERR] flag will set. After the execution of the Mass Erase function, the FCLKDIV register will be reset and the value of register FCLKDIV must be loaded before launching any other command afterwards.

Before invoking the erase-all function, the FSTAT[ACCERR]and FSTAT[FPVIOL] flags must be clear. When invoked the Debugger Mass Erase request will erase all flash memory space regardless of the protection settings. If the post-erase verify passes, the routine will then release security by setting the FSEC[SEC] to the unsecure state. The security byte in the Flash Configuration Field will be programmed to the unsecure state. The status of the Debugger Mass Erase request is reflected in the FCNFG[ERSAREQ]. The FCNFG[ERSAREQ] will be cleared once the operation has completed and the normal FSTAT error reporting will be available as described in the following table.

At the end of the Mass Erase sequence Protection will remain configured as it was before executing the Mass Erase function. If the application requires programming P-Flash after the Mass Erase function completes, the existing protection limits must be taken into account. If protection needs to be disabled the user may need to reset the system right after completing the Mass Erase function.

Table 18-25. Debugger mass erase request error handling

Register	Error Bit	Error Condition
FSTAT	ACCERR	Set if command not available in current mode.
	MGSTAT1	Set if any errors have been encountered during the erase verify operation, or during the program verify operation.
	MGSTAT0	Set if any non-correctable errors have been encountered during the erase verify operation, or during the program verify operation.

18.3.9.9 Erase flash block command

The erase flash block operation will erase all addresses in a flash block.

Table 18-26. Erase flash block command FCCOB requirements

CCOBIX[2:0]	FCCOB parameters	
000	0x09	Global address [23:16] to identify flash block
001	Global address[15:0] in flash block to be erased	

Upon clearing FSTAT[CCIF] to launch the erase flash block command, the memory controller will erase the selected flash block and verify that it is erased. The FSTAT[CCIF] flag will set after the erase flash block operation has completed.

Table 18-27. Erase flash block command error handling

Register	Error Bit	Error Condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 001 at command launch
		Set if command not available in current mode (see Table 18-4)
		Set if an invalid global address [23:16] is supplied[1]
	FPVIOL	Set if an area of the selected flash block is protected
	MGSTAT1	Set if any errors have been encountered during the verify operation[2]
	MGSTAT0	Set if any non-correctable errors have been encountered during the verify operation[2]

1. As defined by the memory map for NVM.
2. As found in the memory map for NVM.

18.3.9.10 Erase flash sector command

The erase flash sector operation will erase all addresses in a flash sector.

Table 18-28. Erase flash sector command FCCOB requirements

CCOBIX[2:0]	FCCOB parameters	
000	0x0A	Global address [23:16] to identify flash block to be erased
001	Global address [15:0] anywhere within the sector to be erased. Refer to Overview for the flash sector size	

Upon clearing FSTAT[CCIF] to launch the erase flash sector command, the memory controller will erase the selected flash sector and then verify that it is erased. The FSTAT[CCIF] flag will be set after the erase flash sector operation has completed.

Table 18-29. Erase flash sector command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 001 at command launch
		Set if command not available in current mode (see Table 18-4)

Table continues on the next page...

Table 18-29. Erase flash sector command error handling (continued)

Register	Error bit	Error condition
		Set if an invalid global address [23:16] is supplied.[1] (see Table 18-1)
		Set if a misaligned longword address is supplied (global address [1:0] != 00)
	FPVIOL	Set if the selected flash sector is protected
	MGSTAT1	Set if any errors have been encountered during the verify operation
	MGSTAT0	Set if any non-correctable errors have been encountered during the verify operation

1. As defined by the memory map for NVM

18.3.9.11 Unsecure flash command

The unsecure flash command will erase the entire flash memory space, and if the erase is successful, will release security.

Table 18-30. Unsecure flash command FCCOB requirements

CCOBIX[2:0]	FCCOB parameters	
000	0x0B	Not required

Upon clearing FSTAT[CCIF] to launch the unsecure flash command, the memory controller will erase the entire flash memory space and verify that it is erased. If the memory controller verifies that the entire flash memory space was properly erased, security will be released. If the erase verify is not successful, the unsecure flash operation sets FSTAT[MGSTAT1] and terminates without changing the security state. During the execution of this command (FSTAT[CCIF] = 0), the user must not write to any flash module register. The FSTAT[CCIF] flag is set after the unsecure flash operation has completed.

Table 18-31. Unsecure flash command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 000 at command launch
		Set if command is not available in current mode (see Table 18-4)
	FPVIOL	Set if any area of the flash memory is protected
	MGSTAT1	Set if any errors have been encountered during the verify operation[1]
	MGSTAT0	Set if any non-correctable errors have been encountered during the verify operation[1]

1. As found in the memory map for NVM

18.3.9.12 Verify backdoor access key command

The verify backdoor access key command will execute only if it is enabled by the FSEC[KEYEN] bits. The verify backdoor access key command releases security if user-supplied keys match those stored in the flash security bytes of the flash configuration field. See Table 18-1 for details. The code that performs verifying backdoor access command must be running from RAM.

Table 18-32. Verify backdoor access key command FCCOB requirements

CCOBIX[2:0]	FCCOBHI parameters	FCCOBLO parameters
000	0x0C	Not required
001	Key 0	
010	Key 1	
011	Key 2	
100	Key 3	

Upon clearing FSTAT[CCIF] to launch the verify backdoor access key command, the memory controller will check the FSEC[KEYEN] bits to verify that this command is enabled. If not enabled, the memory controller sets the FSTAT[ACCERR] bit. If the command is enabled, the memory controller compares the key provided in FCCOB to the backdoor comparison key in the flash configuration field with Key 0 compared to 0x0400, and so on. If the backdoor keys match, security will be released. If the backdoor keys do not match, security is not released and all future attempts to execute the verify backdoor access key command are aborted (set FSTAT[ACCERR]) until a reset occurs. The FSTAT[CCIF] flag is set after the verify backdoor access key operation has completed.

Table 18-33. Verify backdoor access key command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] ≠ 100 at command launch
		Set if an incorrect backdoor key is supplied
		Set if backdoor key access has not been enabled (KEYEN[1:0] ≠ 10
		Set if the backdoor key has mismatched since the last reset
	FPVIOL	None
	MGSTAT1	None
	MGSTAT0	None

18.3.9.13 Set user margin level command

The user margin is a small delta to the normal read reference level and, in effect, is a minimum safety margin. That is, if the reads pass at the tighter tolerances of the user margins, the normal reads have at least that much safety margin before users experience data loss.

The set user margin level command causes the memory controller to set the margin level for future read operations of the flash block.

Table 18-34. Set user margin level command FCCOB requirements

CCOBIX[2:0]	FCCOBHI parameters	FCCOBLO parameters
000	0x0D	Global address [23:16] to identify flash block
001	Global address [15:0] to identify flash block	
010	Margin level setting	

Upon clearing FSTAT[CCIF] to launch the set user margin level command, the memory controller will set the user margin level for the targeted block and then set the FSTAT[CCIF] flag.

Note

Valid margin level settings for the set user margin level command are defined in the following tables.

Table 18-35. Valid set user margin level settings

CCOB (CCOBIX = 010)	Level description
0x0000	Return to normal level
0x0001	User margin-1 level[1]
0x0002	User margin-0 level[2]

1. Read margin to the erased state
2. Read margin to the programmed state

Table 18-36. Set user margin level command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 010 at command launch
		Set if command is not available in current mode (see Table 18-4)
		Set if an invalid global address [23:0] is supplied
		Set if an invalid margin level setting is supplied
	FPVIOL	None
	MGSTAT1	None
	MGSTAT0	None

Note

User margin levels can be used to check that NVM memory contents have adequate margin for normal level read operations. If unexpected results are encountered when checking NVM memory contents at user margin levels, a potential loss of information has been detected.

18.3.9.14 Set factory margin level command

The set factory margin Level command causes the memory controller to set the margin level specified for future read operations of the flash block.

Table 18-37. Set factory margin level command FCCOB requirements

CCOBIX[2:0]	FCCOBHI parameters	FCCOBLO parameters
000	0x0E	Global address [23:16] to identify flash block
001	Global address [15:0] to identify flash block	
010	Margin level setting	

Upon clearing FSTAT[CCIF] to launch the set factory margin level command, the memory controller will set the factory margin level for the targeted block and then set the FSTAT[CCIF] flag.

Note

Valid margin level settings for the set factory margin level command are defined in the following tables.

Table 18-38. Valid set factory margin level settings

CCOB (CCOBIX = 010)	Level description
0x0000	Return to normal level
0x0001	User margin-1 level[1]
0x0002	User margin-0 level[2]
0x0003	Factory margin-1 level[1]
0x0004	Factory margin-0 level[2]

1. Read margin to the erased state
2. Read margin to the programmed state

Table 18-39. Set factory margin level command error handling

Register	Error bit	Error condition
FSTAT	ACCERR	Set if CCOBIX[2:0] != 010 at command launch
		Set if command is not available in current mode (see Table 18-4)
		Set if an invalid global address [23:0] is supplied
		Set if an invalid margin level setting is supplied
	FPVIOL	None
	MGSTAT1	None
	MGSTAT0	None

CAUTION

Factory margin levels must only be used during verify of the initial factory programming.

Note

Factory margin levels can be used to check that Flash memory contents have adequate margin for data retention at the normal level setting. If unexpected results are encountered when checking flash memory contents at factory margin levels, the fash memory contents must be erased and reprogrammed.

18.4 Memory map and register definition

This section presents a high-level summary of the registers and how they are mapped. The registers can be accessed in 32-bits, 16-bits (aligned on data[31:16] or on data[15:0]) or 8-bits. In the case of the writable registers, the write accesses are forbidden during flash command execution. For more details, see Caution note in Flash memory map.

FTMRE memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_0001	Flash CCOB Index Register (FTMRE_FCCOBIX)	8	R/W	00h	18.4.1/249
4002_0002	Flash Security Register (FTMRE_FSEC)	8	R	Undefined	18.4.2/249
4002_0003	Flash Clock Divider Register (FTMRE_FCLKDIV)	8	R/W	00h	18.4.3/250
4002_0005	Flash Status Register (FTMRE_FSTAT)	8	R/W	80h	18.4.4/251
4002_0007	Flash Configuration Register (FTMRE_FCNFG)	8	R/W	00h	18.4.5/252
4002_0008	Flash Common Command Object Register: Low (FTMRE_FCCOBLO)	8	R/W	00h	18.4.6/253

Table continues on the next page...

FTMRE memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4002_0009	Flash Common Command Object Register:High (FTMRE_FCCOBHI)	8	R/W	00h	18.4.7/254
4002_000B	Flash Protection Register (FTMRE_FPROT)	8	R	See section	18.4.8/254
4002_000F	Flash Option Register (FTMRE_FOPT)	8	R	Undefined	18.4.9/255

18.4.1 Flash CCOB Index Register (FTMRE_FCCOBIX)

The FCCOBIX register is used to index the FCCOB register for NVM memory operations.

Address: 4002_0000h base + 1h offset = 4002_0001h

Bit	7	6	5	4	3	2	1	0
Read	0					CCOBIX		
Write	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0

FTMRE_FCCOBIX field descriptions

Field	Description
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CCOBIX	Common Command Register Index Selects which word of the FCCOB register array is being read or written to.

18.4.2 Flash Security Register (FTMRE_FSEC)

The FSEC register holds all bits associated with the security of the MCU and NVM module. All fields in the FSEC register are readable but not writable. During the reset sequence, the FSEC register is loaded with the contents of the flash security byte in the flash configuration field located in flash memory.

See Security for security function.

Address: 4002_0000h base + 2h offset = 4002_0002h

Bit	7	6	5	4	3	2	1	0
Read	KEYEN		Reserved				SEC	
Write	▨	▨	▨	▨	▨	▨	▨	▨
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

FTMRE_FSEC field descriptions

Field	Description
7–6 KEYEN	Backdoor Key Security Enable Bits The KEYEN[1:0] bits define the enabling of backdoor key access to the flash module. **NOTE:** 01 is the preferred KEYEN state to disable backdoor key access. 00 Disabled 01 Disabled 10 Enabled 11 Disabled
5–2 Reserved	This field is reserved.
SEC	Flash Security Bits Defines the security state of the MCU. If the flash module is unsecured using backdoor key access, the SEC field is forced to 10. **NOTE:** 01 is the preferred SEC state to set MCU to secured state. 00 Secured 01 Secured 10 Unsecured 11 Secured

18.4.3 Flash Clock Divider Register (FTMRE_FCLKDIV)

The FCLKDIV register is used to control timed events in program and erase algorithms.

NOTE

The FCLKDIV register must not be written while a flash command is executing (FSTAT[CCIF] = 0)

Address: 4002_0000h base + 3h offset = 4002_0003h

Bit	7	6	5	4	3	2	1	0
Read	FDIVLD	FDIVLCK	FDIV					
Write	▨							
Reset	0	0	0	0	0	0	0	0

FTMRE_FCLKDIV field descriptions

Field	Description
7 FDIVLD	Clock Divider Loaded

Table continues on the next page...

FTMRE_FCLKDIV field descriptions (continued)

Field	Description
	0 FCLKDIV register has not been written since the last reset. 1 FCLKDIV register has been written since the last reset.
6 FDIVLCK	Clock Divider Locked 0 FDIV field is open for writing. 1 FDIV value is locked and cannot be changed. After the lock bit is set high, only reset can clear this bit and restore writability to the FDIV field in user mode.
FDIV	Clock Divider Bits FDIV[5:0] must be set to effectively divide BUSCLK down to 1MHz to control timed events during flash program and erase algorithms. Refer to the table in the Writing the FCLKDIV register for the recommended values of FDIV based on the BUSCLK frequency.

18.4.4 Flash Status Register (FTMRE_FSTAT)

The FSTAT register reports the operational status of the flash module.

Address: 4002_0000h base + 5h offset = 4002_0005h

Bit	7	6	5	4	3	2	1	0
Read	CCIF	0	ACCERR	FPVIOL	MGBUSY	0	MGSTAT	
Write	CCIF	▨	ACCERR	FPVIOL	▨	▨	▨	▨
Reset	1	0	0	0	0	0	0	0

FTMRE_FSTAT field descriptions

Field	Description
7 CCIF	Command Complete Interrupt Flag Indicates that a flash command has completed. The CCIF flag is cleared by writing a 1 to CCIF to launch a command and CCIF will stay low until command completion or command violation. 0 Flash command is in progress. 1 Flash command has completed.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 ACCERR	Flash Access Error Flag Indicates an illegal access has occurred to the flash memory caused by either a violation of the command write sequence or issuing an illegal flash command. While ACCERR is set, the CCIF flag cannot be cleared to launch a command. Writing 1 to this field clears it while writing a 0 to this field has no effect. 0 No access error is detected. 1 Access error is detected.

Table continues on the next page...

FTMRE_FSTAT field descriptions (continued)

Field	Description
4 FPVIOL	Flash Protection Violation Flag Indicates an attempt was made to program or erase an address in a protected area of flash memory during a command write sequence. Writing 1 to FPVIOL clears this field while writing 0 to this field has no effect. While FPIOL is set, it is not possible to launch a command or start a command write sequence. 0 No protection violation is detected. 1 Protection violation is detected.
3 MGBUSY	Memory Controller Busy Flag Reflects the active state of the memory controller. 0 Memory controller is idle. 1 Memory controller is busy executing a flash command (CCIF = 0).
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MGSTAT	Memory Controller Command Completion Status Flag One or more MGSTAT flag bits are set if an error is detected during execution of a flash command or during the flash reset sequence. **NOTE:** Reset value can deviate from the value shown if a double bit fault is detected during the reset sequence.

18.4.5 Flash Configuration Register (FTMRE_FCNFG)

The FCNFG register enables the flash command complete interrupt and forces ECC faults on flash array read access from the CPU.

Address: 4002_0000h base + 7h offset = 4002_0007h

Bit	7	6	5	4	3	2	1	0
Read	CCIE	0	ERSAREQ	0				
Write		▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

FTMRE_FCNFG field descriptions

Field	Description
7 CCIE	Command Complete Interrupt Enable Controls interrupt generation when a flash command has completed. 0 Command complete interrupt is disabled. 1 An interrupt will be requested whenever the CCIF flag in the FSTAT register is set.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FTMRE_FCNFG field descriptions (continued)

Field	Description
5 ERSAREQ	Debugger Mass Erase Request Requests the MGATE to execute the Erase All Blocks command and release security. ERSAREQ is not directly writable but is under indirect user control. The ERSAREQ field sets to 1 when the MGATE starts executing the sequence. ERSAREQ will be reset to 0 by the MGATE when the operation is completed 0 No request or request complete 1 Request to • run the Erase All Blocks command • verify the erased state • program the security byte in the Flash Configuration Field to the unsecure state • release MCU security by setting FSEC[SEC] to the unsecure state
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

18.4.6 Flash Common Command Object Register: Low (FTMRE_FCCOBLO)

The FCCOB is an array of six words addressed via the CCOBIX index found in the FCCOBIX register. Byte-wide reads and writes are allowed to the FCCOB register.

Address: 4002_0000h base + 8h offset = 4002_0008h

Bit	7	6	5	4	3	2	1	0
Read Write	CCOB							
Reset	0	0	0	0	0	0	0	0

FTMRE_FCCOBLO field descriptions

Field	Description
CCOB	Common Command Object Bit 7:0 Low 8 bits of Common Command Object register

18.4.7 Flash Common Command Object Register:High (FTMRE_FCCOBHI)

The FCCOB is an array of six words addressed via the CCOBIX index found in the FCCOBIX register. Byte-wide reads and writes are allowed to the FCCOB register.

Address: 4002_0000h base + 9h offset = 4002_0009h

Bit	7	6	5	4	3	2	1	0
Read / Write	CCOB							
Reset	0	0	0	0	0	0	0	0

FTMRE_FCCOBHI field descriptions

Field	Description
CCOB	Common Command Object Bit 15:8 High 8 bits of Common Command Object register

18.4.8 Flash Protection Register (FTMRE_FPROT)

The FPROT register defines which flash sectors are protected against program and erase operations.

The unreserved bits of the FPROT register are writable with the restriction that the size of the protected region can only be increased (see Protection).

During the reset sequence, the FPROT register is loaded with the contents of the flash protection byte in the flash configuration field at global address 0x40D located in flash memory. To change the flash protection that will be loaded during the reset sequence, the upper sector of the flash memory must be unprotected, then the flash protection byte must be reprogrammed.

Trying to alter data in any protected area in the flash memory will result in a protection violation error and the FPVIOL bit will be set in the FSTAT register. The block erase of a flash block is not possible if any of the flash sectors contained in the same flash block are protected.

Address: 4002_0000h base + Bh offset = 4002_000Bh

Bit	7	6	5	4	3	2	1	0
Read	FPOPEN	RNV6	RNV			FPLDIS	FPLS	
Write		▨	▨	▨	▨			
Reset	*	*	0	0	0	*	*	*

* Notes:

•

FTMRE_FPROT field descriptions

Field	Description
7 FPOPEN	Flash Protection Operation Enable The FPOPEN bit determines the protection function for program or erase operations. 0 When FPOPEN is clear, the FPLDIS field defines unprotected address ranges as specified by the corresponding FPLS field. 1 When FPOPEN is set, the FPLDIS field enables protection for the address range specified by the corresponding FPLS field.
6 RNV6	Reserved Nonvolatile Bit The RNV bit must remain in the erased state.
5–3 RNV	Reserved Nonvolatile Bit The RNV bit must remain in the erased state.
2 FPLDIS	Flash Protection Lower Address Range Disable The FPLDIS bit determines whether there is a protected/unprotected area in a specific region of the flash memory beginning with global address 0x0_0000. 0 Protection/Unprotection enabled. 1 Protection/Unprotection disabled.
FPLS	Flash Protection Lower Address Size The FPLS bits determine the size of the protected/unprotected area in flash memory. The FPLS bits can only be written to while the FPLDIS bit is set.

18.4.9 Flash Option Register (FTMRE_FOPT)

The FOPT register is the flash option register.

During the reset sequence, the FOPT register is loaded from the flash nonvolatile byte in the flash configuration field at global address 0x040F located in flash memory as indicated by reset condition.

Address: 4002_0000h base + Fh offset = 4002_000Fh

Bit	7	6	5	4	3	2	1	0
Read	NV							
Write	▨							
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FTMRE_FOPT field descriptions

Field	Description
NV	Nonvolatile Bits The NV[7:0] bits are available as nonvolatile bits. During the reset sequence, the FOPT register is loaded from the flash nonvolatile byte in the flash configuration field at global address 0x40F located in flash memory.

Chapter 19
Flash Memory Controller (FMC)

19.1 Introduction

The Flash Memory Controller (FMC) is a memory acceleration unit. A list of features provided by the FMC can be found here.

- an interface between bus masters and the 32-bit program flash memory.
- a buffer and a cache that can accelerate program flash memory data transfers.

19.1.1 Overview

The Flash Memory Controller manages the interface between bus masters and the 32-bit program flash memory. The FMC receives status information detailing the configuration of the flash memory and uses this information to ensure a proper interface. The FMC supports 8-bit, 16-bit, and 32-bit read operations from the program flash memory. A write operation to program flash memory results in a bus error.

In addition, the FMC provides two separate mechanisms for accelerating the interface between bus masters and program flash memory. A 32-bit speculation buffer can prefetch the next 32-bit flash memory location, and a 4-way, 2-set program flash memory cache can store previously accessed program flash memory data for quick access times.

19.1.2 Features

The features of FMC module include:

- Interface between bus masters and the 32-bit program flash memory:
    - 8-bit, 16-bit, and 32-bit read operations to nonvolatile flash memory.
- Acceleration of data transfer from the program flash memory to the device:

- 32-bit prefetch speculation buffer for program flash accesses with controls for instruction/data access
- 4-way, 2-set, 32-bit line size program flash memory cache for a total of eight 32-bit entries with invalidation control

19.2 Modes of operation

The FMC operates only when a bus master accesses the program flash memory.

In terms of chip power modes:

- The FMC operates only in Run and Wait modes.
- For any power mode where the program flash memory cannot be accessed, the FMC is disabled.

19.3 External signal description

The FMC has no external (off-chip) signals.

19.4 Memory map and register descriptions

The MCM's programming model provides control and configuration of the FMC's features.

For details, see the description of the MCM's Platform Control Register (PLACR).

19.5 Functional description

The FMC is a flash acceleration unit with flexible buffers for user configuration.

Besides managing the interface between bus masters and the program flash memory, the FMC can be used to customize the program flash memory cache and buffer to provide single-cycle system clock data access times. Whenever a hit occurs for the prefetch speculation buffer or the cache (when enabled), the requested data is transferred within a single system clock.

Upon system reset, the FMC is configured as follows:

- Flash cache is enabled.
- Instruction speculation and caching are enabled.

- Data speculation is disabled.
- Data caching is enabled.

Though the default configuration provides flash acceleration, advanced users may desire to customize the FMC buffer configurations to maximize throughput for their use cases. For example, the user may adjust the controls to enable buffering per access type (data or instruction).

NOTE

When reconfiguring the FMC, do not program the control and configuration inputs to the FMC while the program flash memory is being accessed. Instead, change them with a routine executing from RAM in supervisor mode.

Chapter 20
Internal Clock Source (ICS)

20.1 Introduction

The internal clock source (ICS) module provides clock source choices for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock. The module can provide this FLL clock or either of the internal or external reference clocks as a source for the MCU system clock. There are also signals provided to control a low-power oscillator (OSC) module. These signals configure and enable the OSC module to generate its external crystal/ resonator clock (OSC_OUT) used by peripheral modules and as the ICS external reference clock source. The ICS external reference clock can be the external crystal/ resonator (OSC_OUT) supplied by an OSC, or it can be another external clock source.

The ICS clock source chosen is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived.

20.1.1 Features

The key features of the ICS module are given below:

- Frequency-locked loop (FLL) is trimmable for accuracy
- Internal or external reference clocks can be used to control the FLL.
- Reference divider is provided for external clock.
- Internal reference clock has 9 trim bits available.
- Internal or external reference clocks can be selected as the clock source for the MCU.
- Whichever clock is selected as the source can be divided down.
    - 3-bit select for clock divider is provided
    - Allowable dividers are: 1, 2, 4, 8, 16, 32, 64, 128 if OSC_CR[RANGE] = 0; 32, 64, 128, 256, 512, 1024 if OSC_CR[RANGE] = 1.
- FLL Engaged Internal mode is automatically selected out of reset.
- Selectable digitally-controlled oscillators (DCO) optimized frequency ranges
- FLL lock detector and external clock monitor

  - FLL lock detector with interrupt capability
  - External reference clock monitor with reset capability
- Digitally controlled oscillators optimized for 40-48 MHz frequency range

20.1.2 Block diagram

The following figure is the ICS block diagram.

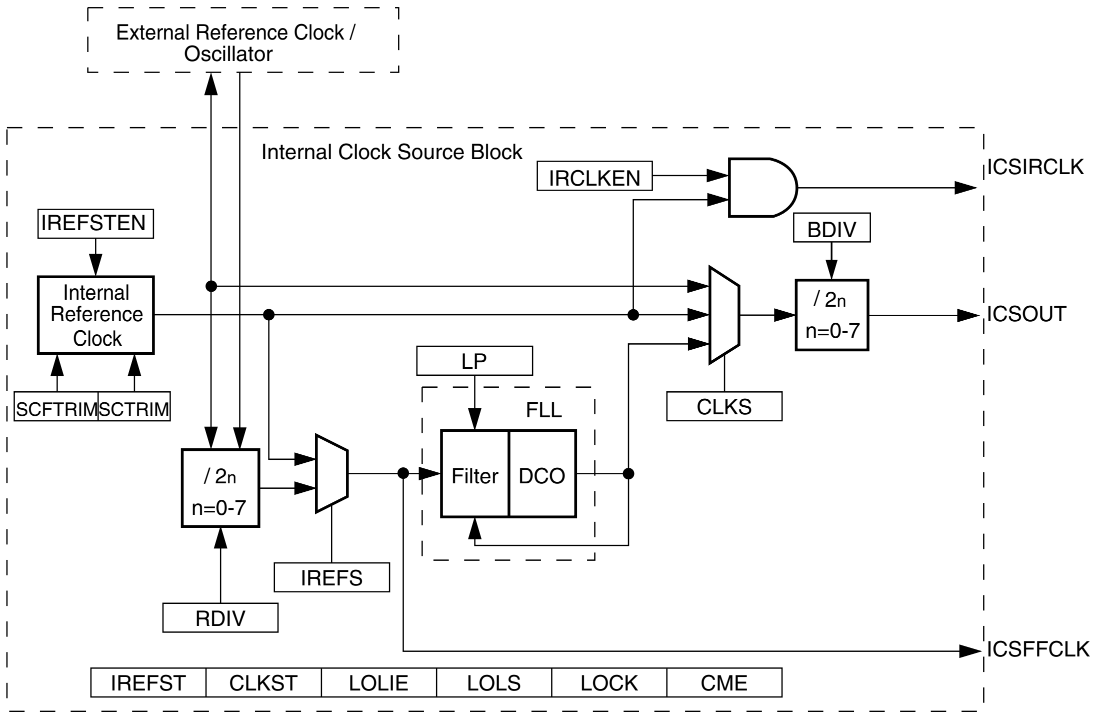


Figure 20-1. Internal clock source (ICS) block diagram

20.1.3 Modes of operation

There are seven modes of operation for the ICS: FEI, FEE, FBI, FBILP, FBE, FBELP, and STOP. Each of these modes is explained briefly in the following subsections.

20.1.3.1 FLL engaged internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from the FLL which is controlled by the internal reference clock.

20.1.3.2 FLL engaged external (FEE)

In FLL engaged external mode, the ICS supplies a clock derived from the FLL which is controlled by an external reference clock source.

20.1.3.3 FLL bypassed internal (FBI)

In FLL bypassed internal mode, the FLL is enabled and controlled by the internal reference clock, but is bypassed. The ICS supplies a clock derived from the internal reference clock.

20.1.3.4 FLL bypassed internal low power (FBILP)

In FLL bypassed internal low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the internal reference clock.

20.1.3.5 FLL bypassed external (FBE)

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The ICS supplies a clock derived from the external reference clock source.

20.1.3.6 FLL bypassed external low power (FBELP)

In FLL bypassed external low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the external reference clock.

20.1.3.7 Stop (STOP)

In Stop mode, the FLL is disabled and the internal or the ICS external reference clocks source (OSC_OUT) can be selected to be enabled or disabled. The ICS does not provide any MCU clock sources.

NOTE

The DCO frequency changes from the pre-stop value to its reset value and the FLL needs to reacquire the lock before the frequency is stable. Timing sensitive operations must wait for the FLL acquisition time, $t_{Acquire}$, before executing.

20.2 External signal description

There are no ICS signals that connect off chip.

20.3 Register definition

ICS memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_4000	ICS Control Register 1 (ICS_C1)	8	R/W	04h	20.3.1/264
4006_4001	ICS Control Register 2 (ICS_C2)	8	R/W	20h	20.3.2/265
4006_4002	ICS Control Register 3 (ICS_C3)	8	R/W	Undefined	20.3.3/266
4006_4003	ICS Control Register 4 (ICS_C4)	8	R/W	See section	20.3.4/267
4006_4004	ICS Status Register (ICS_S)	8	R	10h	20.3.5/267

20.3.1 ICS Control Register 1 (ICS_C1)

Address: 4006_4000h base + 0h offset = 4006_4000h

Bit	7	6	5	4	3	2	1	0
Read / Write	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
Reset	0	0	0	0	0	1	0	0

ICS_C1 field descriptions

Field	Description
7–6 CLKS	Clock Source Select Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of ICS_C2[BDIV]. 00 Output of FLL is selected. 01 Internal reference clock is selected. 10 External reference clock is selected. 11 Reserved, defaults to 00.
5–3 RDIV	Reference Divider Selects the amount to divide down the FLL reference clock selected by the IREFS bits. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz.

Table continues on the next page...

ICS_C1 field descriptions (continued)

Field	Description
	(RDIV table below)

RDIV	OSC_CR[RANGE]= 0	OSC_CR[RANGE]= 1
000	1[1]	32
001	2	64
010	4	128
011	8	256
100	16	512
101	32	1024
110	64	Reserved
111	128	Reserved

1. Reset default

Field	Description
2 IREFS	Internal Reference Select Selects the reference clock source for the FLL. 0 External reference clock is selected. 1 Internal reference clock is selected.
1 IRCLKEN	Internal Reference Clock Enable Enables the internal reference clock for use as ICSIRCLK. 0 ICSIRCLK is inactive. 1 ICSIRCLK is active.
0 IREFSTEN	Internal Reference Stop Enable Controls whether or not the internal reference clock remains enabled when the ICS enters Stop mode. 0 Internal reference clock is disabled in Stop mode. 1 Internal reference clock stays enabled in Stop mode if IRCLKEN is set, or if ICS is in FEI, FBI, or FBILP mode before entering Stop.

1. Reset default

20.3.2 ICS Control Register 2 (ICS_C2)

Address: 4006_4000h base + 1h offset = 4006_4001h

Bit	7	6	5	4	3	2	1	0
Read	BDIV			LP	0			
Write					▨	▨	▨	▨
Reset	0	0	1	0	0	0	0	0

ICS_C2 field descriptions

Field	Description
7–5 BDIV	Bus Frequency Divider

Table continues on the next page...

ICS_C2 field descriptions (continued)

Field	Description
	Selects the amount to divide down the clock source selected by ICS_C1[CLKS]. This controls the bus frequency. 000 Encoding 0—Divides the selected clock by 1. 001 Encoding 1—Divides the selected clock by 2 (reset default). 010 Encoding 2—Divides the selected clock by 4. 011 Encoding 3—Divides the selected clock by 8. 100 Encoding 4—Divides the selected clock by 16. 101 Encoding 5—Divides the selected clock by 32. 110 Encoding 6—Divides the selected clock by 64. 111 Encoding 7—Divides the selected clock by 128.
4 LP	Low Power Select Controls whether the FLL is disabled in FLL bypassed modes. 0 FLL is not disabled in bypass mode. 1 FLL is disabled in bypass modes unless debug is active.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

20.3.3 ICS Control Register 3 (ICS_C3)

Address: 4006_4000h base + 2h offset = 4006_4002h

Bit	7	6	5	4	3	2	1	0
Read Write	SCTRIM							
Reset	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

ICS_C3 field descriptions

Field	Description
SCTRIM	Slow Internal Reference Clock Trim Setting Controls the slow internal reference clock frequency by controlling the internal reference clock period. The bits are binary weighted. In other words, bit 1 adjusts twice as much as bit 0. Increasing the binary value in SCTRIM will increase the period, and decreasing the value will decrease the period. An additional fine trim bit is available as the ICS_C4[SCFTRIM]. SCTRIM is loaded during reset from a factory programmed location if the device is not in any debug mode.

20.3.4 ICS Control Register 4 (ICS_C4)

Address: 4006_4000h base + 3h offset = 4006_4003h

Bit	7	6	5	4	3	2	1	0
Read	LOLIE	0	CME	0				SCFTRIM
Write		▨		▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	x*

* Notes:
- x = Undefined at reset.

ICS_C4 field descriptions

Field	Description
7 LOLIE	Loss of Lock Interrupt Determines if an interrupt request is made following a loss of lock indication. This field has an effect only when ICS_S[LOLS] is set. 0 No request on loss of lock. 1 Generates an interrupt request on loss of lock.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 CME	Clock Monitor Enable Determines if a reset request is made following a loss of external clock indication. This field must be set to a logic 1 only when the ICS is in an operational mode that uses the external clock (FEE, FBE, or FBELP). 0 Clock monitor is disabled. 1 Generates a reset request on loss of external clock.
4–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 SCFTRIM	Slow Internal Reference Clock Fine Trim Controls the smallest adjustment of the internal reference clock frequency. Setting SCFTRIM will increase the period and clearing SCFTRIM will decrease the period by the smallest amount possible. **NOTE:** SCFTRIM is loaded during reset from a factory programmed location when not in any debug mode.

20.3.5 ICS Status Register (ICS_S)

Address: 4006_4000h base + 4h offset = 4006_4004h

Bit	7	6	5	4	3	2	1	0
Read	LOLS	LOCK	0	IREFST	CLKST		0	
Write	w1c	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	1	0	0	0	0

ICS_S field descriptions

Field	Description
7 LOLS	Loss of Lock Status Indicates the lock status for the FLL. LOLS is set when lock detection is enabled and after acquiring lock, the FLL output frequency has fallen outside the lock exit frequency tolerance, from ±4.7% to ±5.97%. ICS_C4[LOLIE] determines whether an interrupt request is made when set. LOLS is cleared by reset or by writing a logic 1 to LOLS when LOLS is set. Writing a logic 0 to LOLS has no effect. 0 FLL has not lost lock since LOLS was last cleared. 1 FLL has lost lock since LOLS was last cleared.
6 LOCK	Lock Status Indicates whether the FLL has acquired lock. Lock detection is disabled when FLL is disabled. If the lock status bit is set then changing the value of any of the following fields IREFS, RDIV[2:0], or, if in FEI or FBI modes, TRIM[7:0] will cause the lock status bit to clear and stay cleared until the FLL has reacquired lock. Stop mode entry will also cause the lock status bit to clear and stay cleared until the FLL has reacquired lock. 0 FLL is currently unlocked. 1 FLL is currently locked.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 IREFST	Internal Reference Status Indicates the current source for the reference clock. This field does not update immediately after a write to ICS_C1[IREFS] due to internal synchronization between clock domains. 0 Source of reference clock is external clock. 1 Source of reference clock is internal clock.
3–2 CLKST	Clock Mode Status Indicates the current clock mode. This field doesn't update immediately after a write to ICS_C1[CLKS] due to internal synchronization between clock domains. 00 Output of FLL is selected. 01 FLL Bypassed, internal reference clock is selected. 10 FLL Bypassed, external reference clock is selected. 11 Reserved.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

20.4 Functional description

20.4.1 Operational modes

The seven states of the ICS are shown as a state diagram and are described below. The arrows indicate the allowed movements among the states.

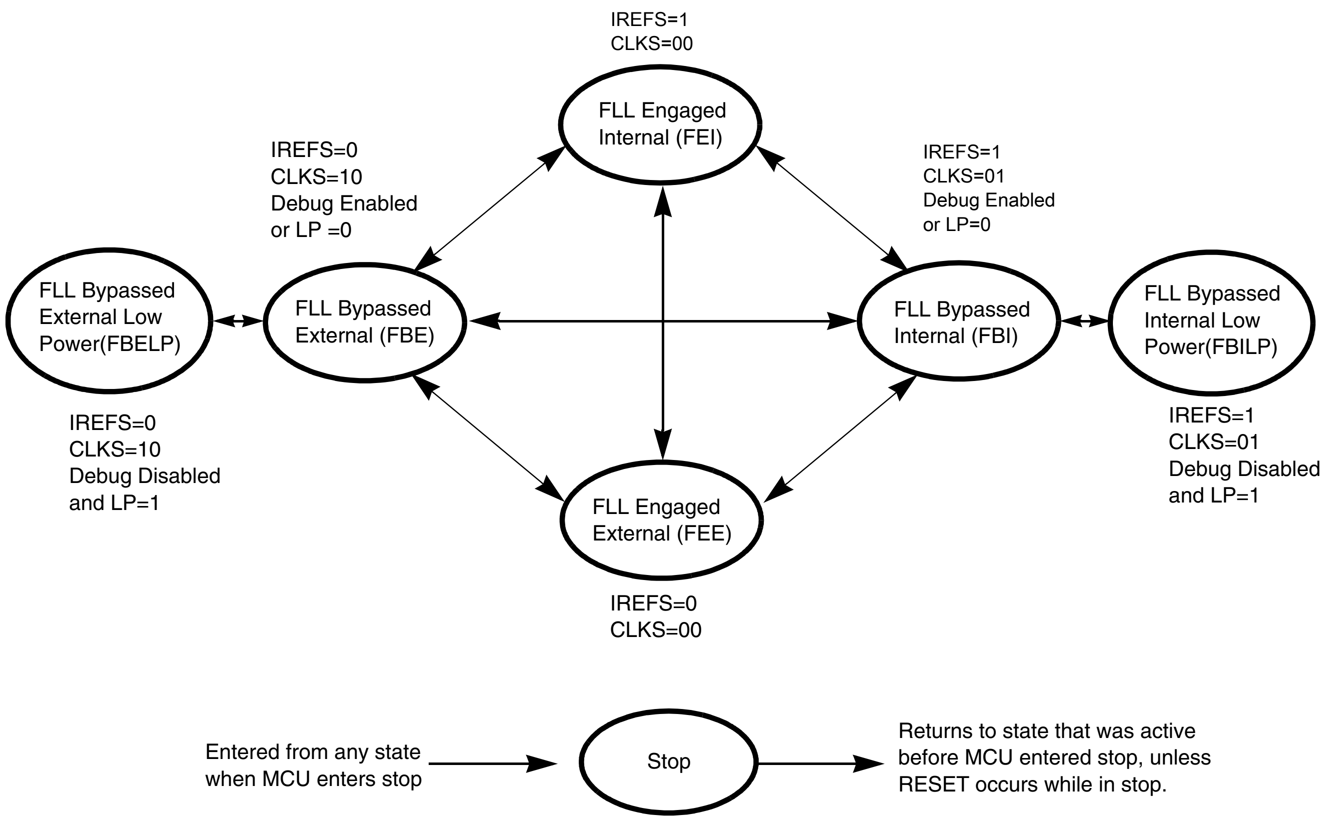

IREFS=1
CLKS=00
FLL Engaged Internal (FEI)
IREFS=0
CLKS=10
Debug Enabled
or LP =0
IREFS=1
CLKS=01
Debug Enabled
or LP=0
FLL Bypassed External Low Power(FBELP)
FLL Bypassed External (FBE)
FLL Bypassed Internal (FBI)
FLL Bypassed Internal Low Power(FBILP)
IREFS=0
CLKS=10
Debug Disabled
and LP=1
IREFS=1
CLKS=01
Debug Disabled
and LP=1
FLL Engaged External (FEE)
IREFS=0
CLKS=00
Entered from any state when MCU enters stop
Stop
Returns to state that was active before MCU entered stop, unless RESET occurs while in stop.

Figure 20-7. Clock switching modes

20.4.1.1 FLL engaged internal (FEI)

FLL engaged internal (FEI) is the default mode of operation and is entered when all the following conditions occur:

- 00b is written to ICS_C1[CLKS].
- 1b is written to ICS_C1[IREFS].

In FLL engaged internal mode, the ICSOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL loop locks the frequency to 1280 times the internal reference frequency. The internal reference clock is enabled.

20.4.1.2 FLL engaged external (FEE)

The FLL engaged external (FEE) mode is entered when all the following conditions occur:

- 00b is written to ICS_C1[CLKS].
- 0b is written to ICS_C1[IREFS].
- ICS_C1[RDIV] and and OSC_CR[RANGE] are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.

In FLL engaged external mode, the ICSOUT clock is derived from the FLL clock which is controlled by the external reference clock source.The FLL loop locks the frequency to 1280 times the external reference frequency, as selected by ICS_C1[RDIV] and OSC_CR[RANGE]. The external reference clock is enabled.

20.4.1.3 FLL bypassed internal (FBI)

The FLL bypassed internal (FBI) mode is entered when all the following conditions occur:

- 01b is written to ICS_C1[CLKS].
- 1b is written to ICS_C1[IREFS].

In FLL bypassed internal mode, the ICSOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL loop locks the FLL frequency to 1280 times the internal reference frequency. The internal reference clock is enabled.

20.4.1.4 FLL bypassed internal low power (FBILP)

The FLL bypassed internal low power (FBILP) mode is entered when all the following conditions occur:

- 01b is written to ICS_C1[CLKS].
- 1b is written to ICS_C1[IREFS].

In FLL bypassed internal low-power mode, the ICSOUT clock is derived from the internal reference clock and the FLL is disabled. The internal reference clock is enabled.

20.4.1.5 FLL bypassed external (FBE)

The FLL bypassed external (FBE) mode is entered when all the following conditions occur:

- 10b is written to ICS_C1[CLKS].
- 0b is written to ICS_C1[IREFS].
- ICS_C1[RDIV] and OSC_CR[RANGE] fields are written to divide external reference clock to be within the range of 31.25 kHz to 39.0625 kHz.

In FLL bypassed external mode, the ICSOUT clock is derived from the external reference clock source. The FLL clock is controlled by the external reference clock, and the FLL loop locks the FLL frequency to 1280 times the external reference frequency, as selected by ICS_C1[RDIV] and OSC_CR[RANGE].

20.4.1.6 FLL bypassed external low power (FBELP)

The FLL bypassed external low-power (FBELP) mode is entered when all the following conditions occur:

- 10b is written to ICS_C1[CLKS].
- 0b is written to ICS_C1[IREFS].

In FLL bypassed external low-power mode, the ICSOUT clock is derived from the external reference clock source and the FLL is disabled. The external reference clock source is enabled.

20.4.1.7 Stop

NOTE

The DCO frequency changes from the pre-stop value to its reset value and the FLL need to re-acquire the lock before the frequency is stable. Timing sensitive operations must wait for the FLL acquisition time, $t_{Acquire}$, before executing.

Stop mode is entered whenever the MCU enters a STOP state. In this mode, all ICS clock signals are static except in the following cases:

ICSIRCLK will be active in Stop mode when all the following conditions occur:

- 1b is written to ICS_C1[IRCLKEN].
- 1b is written to ICS_C1[IREFSTEN].

20.4.2 Mode switching

ICS_C1[IREFS] can be changed at anytime, but the actual switch to the newly selected clock is shown by ICS_S[IREFST]. When switching between FLL engaged internal (FEI) and FLL engaged external (FEE) modes, the FLL begins locking again after the switch is completed.

ICS_C1[CLKS] can also be changed at anytime, but the actual switch to the newly selected clock is shown by ICS_S[CLKST]. If the newly selected clock is not available, the previous clock remains selected.

NOTE

When mode switching is from FEE, FBE or FBELP to FEI, it is suggested to wait IREFST switch completion, then change ICS_C1[CLKS].

20.4.3 Bus frequency divider

ICS_C2[BDIV] can be changed anytime and the actual switch to the new frequency occurs immediately.

20.4.4 Low-power field usage

The Low-Power (LP) field in the ICS_C2 register is provided to allow the FLL to be disabled and thus conserve power when it is not being used.

However, in some applications it may be desirable to allow the FLL to be enabled and to lock for maximum accuracy before switching to an FLL engaged mode. To do this, write 0b to ICS_C2[LP].

20.4.5 Internal reference clock

When ICS_C1[IRCLKEN] is set, the internal reference clock signal is presented as ICSIRCLK, which can be used as an additional clock source. To re-target the ICSIRCLK frequency, write a new value to the ICS_C3[SCTRIM] bits to trim the period of the internal reference clock:

- Writing a larger value slows down the ICSIRCLK frequency.
- Writing a smaller value to the ICS_C3 register speeds up the ICSIRCLK frequency.

The TRIM bits effect the ICSOUT frequency if the ICS is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or FLL bypassed internal low power (FBILP) mode.

Until ICSIRCLK is trimmed, programming low bus divider (ICS_C2[BDIV]) factors may result in ICSOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications.

If ICS_C1[IREFSTEN] is set and 1b is written to ICS_C1[IRCLKEN], the internal reference clock keeps running during Stop mode in order to provide a fast recovery upon exiting Stop mode.

All MCU devices are factory programmed with a trim value in a reserved memory location. This value is uploaded to the ICS_C3 register and ICS_C4[SCFTRIM] during any reset initialization. For finer precision, trim the internal oscillator in the application and set ICS_C4[SCFTRIM] accordingly.

20.4.6 Fixed frequency clock

The ICS presents the divided FLL reference clock as ICSFFCLK for use as an additional clock source. ICSFFCLK frequency must be no more than 1/4 of the ICSOUT frequency to be valid. Because of this requirement, in bypass modes, the ICSFFCLK is valid only in bypass external modes (FBE and FBELP) for the following conditions of ICS_C2[BDIV], and divider factor of ICS_C1[RDIV] and OSC_CR[RANGE] values:

if OSC_CR[RANGE] is high,

- ICS_C2[BDIV] = 000, ICS_C2[RDIV] ≥ 010
- ICS_C2[BDIV] = 001 (divide by 2), ICS_C2[RDIV] ≥ 011
- ICS_C2[BDIV] = 010 (divide by 4), ICS_C2[RDIV] ≥ 100
- ICS_C2[BDIV] = 011 (divide by 8), ICS_C2[RDIV] ≥ 101

20.4.7 FLL lock and clock monitor

20.4.7.1 FLL clock lock

In FBE and FEE modes, the clock detector source uses the external reference as the reference. When FLL is detected from lock to unlock, ICS_S[LOLS] is set. An interrupt will be generated if ICS_C4[LOLIE] is set. ICS_S[LOLS] is cleared by reset or by writing a logic 1 to ICS_S[LOLS] when ICS_S[LOLS] is set. Writing a logic 0 to ICS_S[LOLS] has no effect.

In FBI and FEI modes, the lock detector source uses the internal reference as the reference. When FLL is detected from lock to unlock, ICS_S[LOLS] is set. An interrupt will be generated if ICS_C4[LOLIE] is set. ICS_S[LOLS] is cleared by reset or by writing a logic 1 to ICS_S[LOLS] when ICS_S[LOLS] is set. Writing a logic 0 to ICS_S[LOLS] has no effect.

In FBELP and FBILP modes, the FLL is not on so that lock detect function is not applicable.

20.4.7.2 External reference clock monitor

In FBE, FEE, or FBELP modes, if 1 is written to ICS_C4[CME], the clock monitor is enabled. If the external reference falls below a certain frequency, the MCU will reset. The SIM_SRSID[LOC] will be set to indicate the error.

20.5 Initialization/application information

This section provides example code to give some basic direction to a user on how to initialize and configure the ICS module. The example software is implemented in C language.

20.5.1 Initializing FEI mode

The following code segment demonstrates setting ICS to FEI mode.

Example: 20.5.1.1 FEI mode initialization routine

```
/* the following code segment demonstrates setting ICS to FEI mode generating 36MHz bus*/
ICS_C2 = 0x00; /*BDIV=0, no prescalar
ICS_C1 = 0x04; /* internal reference clock to FLL */
ICS_C3 = TRIM_VALUE_35.15625KHZ; /* FLL output 36MHz */
/* the following code segment demonstrates setting ICS to FEI mode generating 4.5MHz bus*/
ICS_C2 = 0x60; /*BDIV=3, prescalar = 8 */
ICS_C1 = 0x04; /* internal reference clock to FLL */
ICS_C3 = TRIM_VALUE_35.15625KHZ; /* FLL output 4.5MHz */
```

20.5.2 Initializing FBI mode

The following code segment demonstrates setting ICS to FBI mode.

Example: 20.5.2.1 FBI mode initialization routine

```
/* the following code segment demonstrates setting ICS to FBI mode generating 32768Hz
bus*/
ICS_C2 = 0x00; /*BDIV=0, no prescalar
ICS_C1 = 0x44;
ICS_C3 = TRIM_VALUE_32K768HZ;
```

20.5.3 Initializing FEE mode

The following code segment demonstrates setting ICS to FEE mode.

Example: 20.5.3.1 FEE mode initialization routine

```
/* the following code segment demonstrates setting ICS to FEE mode generating 32.768MHZ bus*/
/* supposing external 32.768K crystal is installed */
OSC_CR = 0xB0; /* enable oscillator with low power mode */
ICS_C2 = 0x00; /* BDIV=0, no prescalar/
while ((OSC_CR & OSC_CR_OSCINIT_MASK) == 0); /* waiting until oscillator is ready */
```

```
ICS_C1 = 0x80; /* external clock reference (32.768kHz) to FLL, RDIV = 0, external prescalar
= 0 */
```

20.5.4 Initializing FBE mode

The following code segment demonstrates setting ICS to FBE mode.

Example: 20.5.4.1 FBE mode initialization routine

```
/* the following code segment demonstrates setting ICS to FBE mode generating 20MHZ bus*/
/* supposing external 20MHZ crystal is installed */
OSC_CR = 0xA6; /* enable clock in high range, high gain mode */
ICS_C2 = 0x00; /* BDIV=0, no prescalar*/
ICS_C1 = 0x80; /* external clock reference (20MHZ) to FLL output */
```

Chapter 21
Oscillator (OSC)

21.1 Introduction

21.1.1 Overview

The OSC module provides the clock source for the MCU. The OSC module, in conjunction with an external crystal or resonator, generates a clock for the MCU that can be used as reference clock or bus clock.

21.1.2 Features and modes

Key features of the OSC module are:

- Supports 32 kHz crystals (low range mode)
- Supports 4–24 MHz crystals and resonators (high range mode)
- Automatic gain control (AGC) to optimize power consumption in both frequency ranges using low-power mode (low gain mode)
- High gain option in both frequency ranges: 32 kHz, 4–24 MHz
- Voltage and frequency filtering to guarantee clock frequency and stability
- Supports to be enabled by ICS.

21.1.3 Block diagram

See the following figure for OSC module block diagram.

The OSC module uses a crystal or resonator to generate three filtered oscillator clock signals(XTL_CLK). The XTL_CLK can work in Stop mode since they come from Hard block which always has power.

The OSCOS decides whether OSC_OUT comes from internal oscillators(XTL_CLK) or directly from external clock driven on EXTAL pin. The OSCOS signal allows the XTAL pad to be used as I/O or test clock.

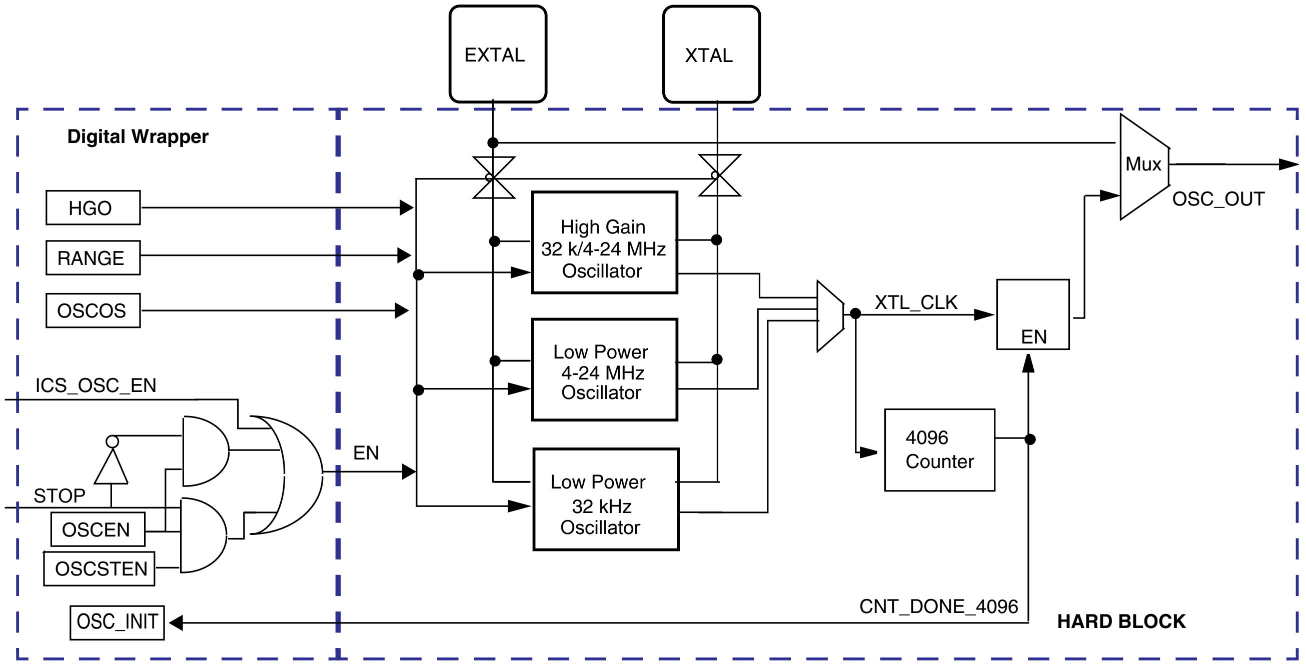


Figure 21-1. OSC module block diagram

21.2 Signal description

The following table shows the user-accessible signals available for the OSC module. See the chip-level specification to find out which signals are actually connected to external pins.

Table 21-1. OSC signal descriptions

Signal	Description	I/O
EXTAL	External clock/oscillator input	Analog input
XTAL	Oscillator output	Analog output

21.3 External crystal / resonator connections

The connections for a crystal/resonator frequency reference are shown in Figure 21-2 and Figure 21-3. When using low-frequency, low-power mode, the only external component is the crystal or resonator itself. In the other oscillator modes, load capacitors (C_x, C_y) and feedback resistor (R_F) are required. In addition, a series resistor (R_S) may be used in high-gain modes. Recommended component values are listed in the data sheet.

Table 21-2. External crystal/resonator connections

Oscillator mode	Connections
Low frequency, high gain	Connection2
Low frequency, low-power	Connection1
High frequency, high gain (4–20 MHz)	Connection2
High frequency, low-power (4–20 MHz)	Connection2

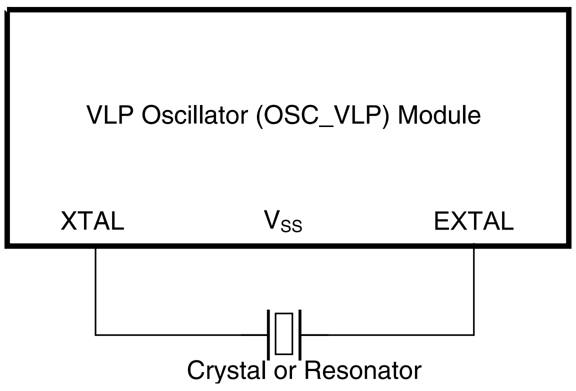


Figure 21-2. Crystal/resonator connections - connection 1

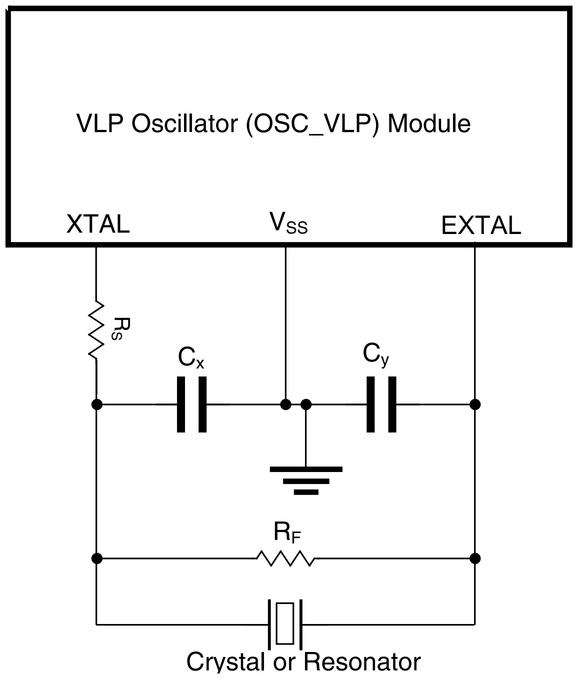


Figure 21-3. Crystal/resonator connections - connection 2

21.4 External clock connections

In external clock mode (OSC_CR[OSCOS] = 0), the pins can be connected as shown in the following figure.

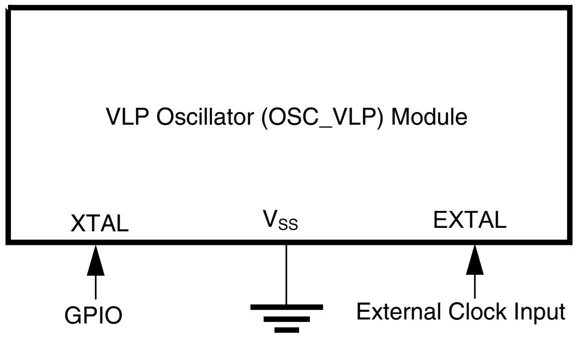


Figure 21-4. External clock connections

21.5 Memory map and register descriptions

OSC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_5000	OSC Control Register (OSC_CR)	8	R/W	00h	21.5.1/281

21.5.1 OSC Control Register (OSC_CR)

Address: 4006_5000h base + 0h offset = 4006_5000h

Bit	7	6	5	4	3	2	1	0
Read	OSCEN	0	OSCSTEN	OSCOS	0	RANGE	HGO	OSCINIT
Write		▨			▨			▨
Reset	0	0	0	0	0	0	0	0

OSC_CR field descriptions

Field	Description
7 OSCEN	OSC Enable Enables the OSC module. The OSC module can also be enabled by the ICS module. 0 OSC module is disabled. 1 OSC module is enabled.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 OSCSTEN	OSC Enable in Stop mode Controls whether or not the OSC clock remains enabled when MCU enters Stop mode and OSCEN is set. OSCSTEN has no effect if ICS requests OSC enable. 0 OSC clock is disabled in Stop mode. 1 OSC clock stays enabled in Stop mode.
4 OSCOS	OSC Output Select Selects the output clock of the OSC module. 0 External clock source from EXTAL pin is selected. 1 Oscillator clock source is selected.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 RANGE	Frequency Range Select Selects the frequency range for the OSC module.

Table continues on the next page...

OSC_CR field descriptions (continued)

Field	Description
	0 Low frequency range of 32 kHz. 1 High frequency range of 4–24 MHz.
1 HGO	High Gain Oscillator Select Controls the OSC mode of operation. 0 Low-power mode 1 High-gain mode
0 OSCINIT	OSC Initialization This field is set after the initialization cycles of oscillator are completed. 0 Oscillator initialization is not complete. 1 Oscillator initialization is completed.

21.6 Functional description

21.6.1 OSC module states

There are three states of the OSC module. A state diagram is shown in Figure 21-6. The states and the transitions among each other are described in this section.

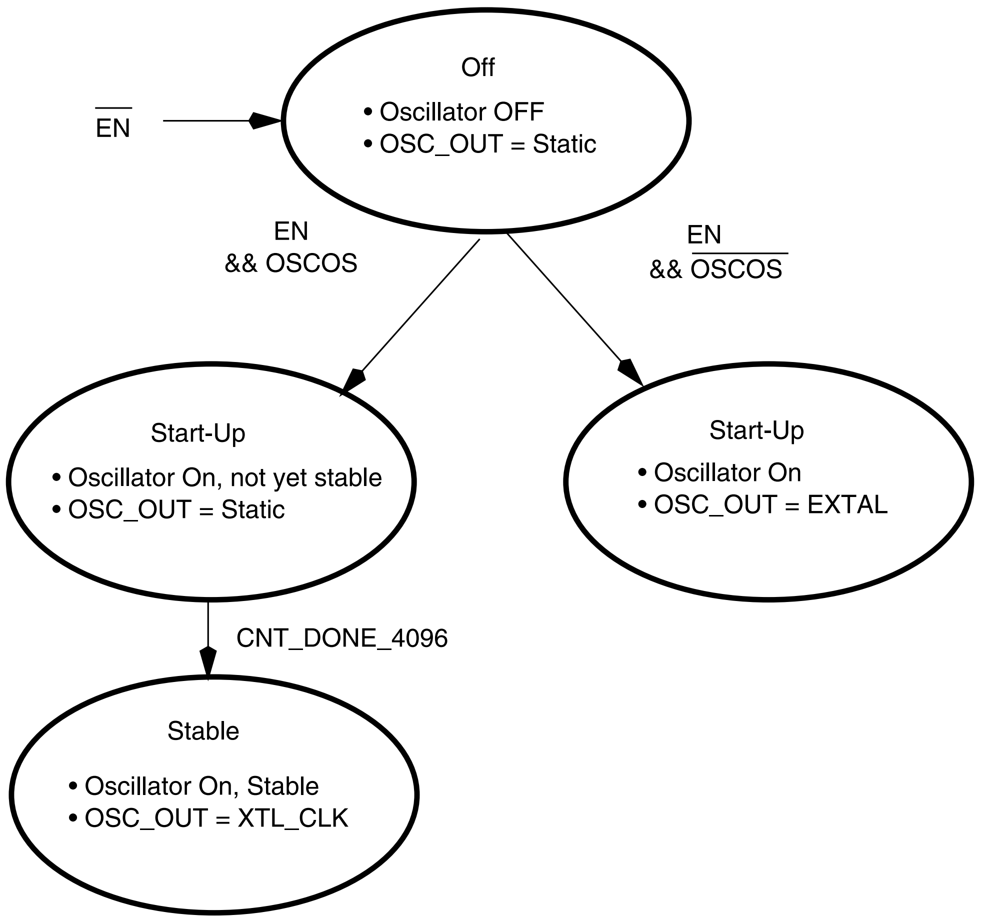


Figure 21-6. OSC module state diagram

EN is decided by OSC_CR[OSCEN], Stop, OSC_CR[OSCSTEN], and external request (ICS_OSC_EN). See the following table for details.

Table 21-5. EN status

EN	ICS_OSC_EN	OSC_CR[OSCEN]	OSC_CR[OSCSTEN]	Stop
1	1	-	-	-
1	0	1	-	0
1	0	1	1	1
0	0	1	0	1
0	0	0	-	-

21.6.1.1 Off

The off state is entered whenever the EN signal is negated. Upon entering this state, XTL_CLK and OSC_OUT is static. The EXTAL and XTAL pins are also decoupled from all other oscillator circuitry in this state. The OSC module circuitry is configured to draw minimal current.

21.6.1.2 Oscillator startup

The oscillator startup state is entered whenever the oscillator is first enabled (EN transitions high) and OSC_CR[OSCOS] is high. In this state, the OSC module is enabled and oscillations are starting up, but have not yet stabilized. When the oscillation amplitude becomes large enough to pass through the input buffer, XTL_CLK begins clocking the counter. When the counter has seen 4096 cycles of XTL_CLK, the oscillator is considered stable and XTL_CLK is passed to the output clock OSC_OUT.

21.6.1.3 Oscillator stable

The oscillator stable state is entered whenever the oscillator is enabled (EN is high), OSC_CR[OSCOS] is high, and the counter has seen 4096 cycles of XTL_CLK (CNT_DONE_4096 is high). In this state, the OSC module is producing a stable output clock on OSC_OUT. Its frequency is determined by the external components being used.

21.6.1.4 External clock mode

The external clock state is entered when the oscillator is enabled(EN is high) and OSC_CR[OSCOS] is low. In this state, the OSC module is set up to buffer (with hysteresis) a clock from EXTAL onto the OSC_OUT. Its frequency is determined by the external clock being supplied.

21.6.2 OSC module modes

The oscillator is a Pierce-type oscillator that supports external crystals or resonators operating over the frequency ranges shown in the following table. These modes assume EN = 1, OSC_CR[OSCOS] = 1.

Table 21-6. Oscillator modes

RANGE	HGO	Mode	Frequency range
0	1	Low-frequency, high-gain	f_{lo}(min) up to f_{lo}(max)
0	0	Low-frequency, low-power (VLP)	
1	1	High-frequency mode1, high-gain	f_{hi}(min) up to f_{hi}(max)
1	0	High-frequency mode1, low-power	

21.6.2.1 Low-frequency, high-gain mode

In low-frequency, high-gain mode (OSC_CR[RANGE] = 0, OSC_CR[HGO] = 1) the oscillator uses a simple inverter-style amplifier. The gain is set to achieve rail-to-rail oscillation amplitudes. The oscillator input buffer in this mode is single-ended. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels.

21.6.2.2 Low-frequency, low-power mode

In low-frequency, low-power mode (OSC_CR[RANGE] = 0, OSC_CR[HGO] = 0), the oscillator uses a gain control loop to minimize power consumption. As the oscillation amplitude increases, the amplifier current is reduced. This continues until a desired amplitude is achieved at steady-state.

The oscillator input buffer in this mode is single-ended. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels.

In this mode, the amplifier inputs, gain-control input, and input buffer input are all capacitively coupled for leakage tolerance (not sensitive to the DC level of EXTAL).

Also in this mode, all external components except for the resonator itself are integrated, which includes the load capacitors and feeback resistor which biases EXTAL.

21.6.2.3 High-frequency, high-gain mode

In high-frequency, high-gain Mode (OSC_CR[RANGE] = 1, OSC_CR[HGO] = 1), the oscillator uses a simple inverter-style amplifier. The gain is set to achieve rail-to-rail oscillation amplitudes.

The oscillator input buffer in this mode is single-ended. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels.

21.6.2.4 High-frequency, low-power mode

In high-frequency, low-power mode (OSC_CR[RANGE] = 1, OSC_CR[HGO] = 0) the oscillator uses a gain control loop to minimize power consumption. As the oscillation amplitude increases, the amplifier current is reduced. This continues until a desired amplitude is achieved at steady-state.

The oscillator input buffer in this mode is differential. It provides low pass frequency filtering as well as hysteresis for voltage filtering and converts the output to logic levels.

21.6.3 Counter

The oscillator output clock (OSC_OUT) is gated off until the counter has detected 4096 cycles of its input clock (XTL_CLK). Once 4096 cycles are complete, the counter passes XTL_CLK onto OSC_OUT. This counting timeout is used to guarantee output clock stability.

21.6.4 Reference clock pin requirements

The OSC module requires use of both the EXTAL and XTAL pins to generate an output clock in oscillator mode but requires only the EXTAL pin in external clock mode. The EXTAL and XTAL pins can be used for I/O or test clock purposes as long as the specifications listed in the data sheet are met.

Chapter 22
Cyclic Redundancy Check (CRC)

22.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The cyclic redundancy check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial, WAS, and other parameters required to implement a 16-bit or 32-bit CRC standard.

The 16/32-bit code is calculated for 32 bits of data at a time.

22.1.1 Features

Features of the CRC module include:

- Hardware CRC generator circuit using a 16-bit or 32-bit programmable shift register
- Programmable initial seed value and polynomial
- Option to transpose input data or output data (the CRC result) bitwise or bytewise. This option is required for certain CRC standards. A bytewise transpose operation is not possible when accessing the CRC data register via 8-bit accesses. In this case, the user's software must perform the bytewise transpose function.
- Option for inversion of final CRC result
- 32-bit CPU register programming interface

22.1.2 Block diagram

The following is a block diagram of the CRC.

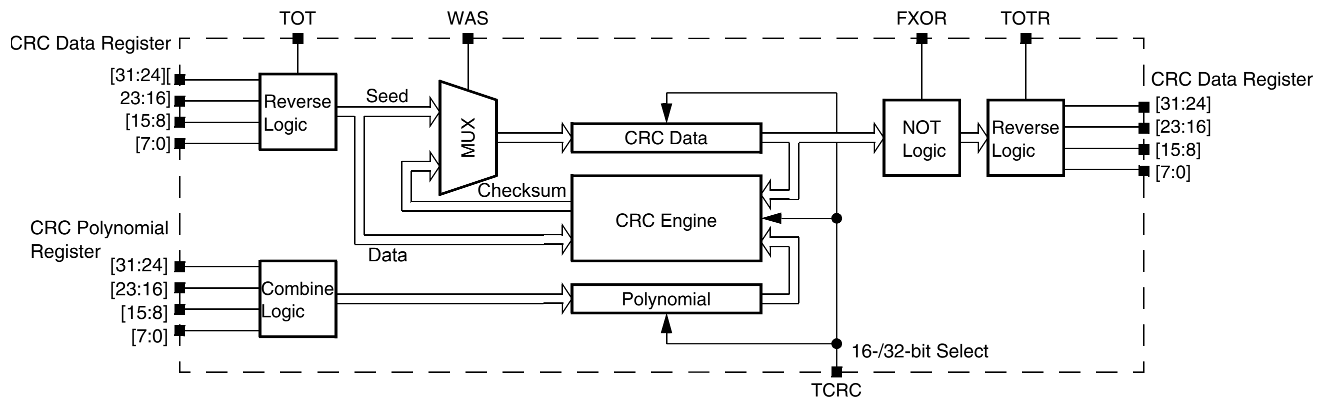


Figure 22-1. Programmable cyclic redundancy check (CRC) block diagram

22.1.3 Modes of operation

Various MCU modes affect the CRC module's functionality.

22.1.3.1 Run mode

This is the basic mode of operation.

22.1.3.2 Low-power modes (Wait or Stop)

Any CRC calculation in progress stops when the MCU enters a low-power mode that disables the module clock. It resumes after the clock is enabled or via the system reset for exiting the low-power mode. Clock gating for this module is dependent on the MCU.

22.2 Memory map and register descriptions

CRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_2000	CRC Data register (CRC_DATA)	32	R/W	FFFF_FFFFh	22.2.1/289
4003_2004	CRC Polynomial register (CRC_GPOLY)	32	R/W	0000_1021h	22.2.2/290
4003_2008	CRC Control register (CRC_CTRL)	32	R/W	0000_0000h	22.2.3/290

22.2.1 CRC Data register (CRC_DATA)

The CRC Data register contains the value of the seed, data, and checksum. When CTRL[WAS] is set, any write to the data register is regarded as the seed value. When CTRL[WAS] is cleared, any write to the data register is regarded as data for general CRC computation.

In 16-bit CRC mode, the HU and HL fields are not used for programming the seed value, and reads of these fields return an indeterminate value. In 32-bit CRC mode, all fields are used for programming the seed value.

When programming data values, the values can be written 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous; with MSB of data value written first.

After all data values are written, the CRC result can be read from this data register. In 16-bit CRC mode, the CRC result is available in the LU and LL fields. In 32-bit CRC mode, all fields contain the result. Reads of this register at any time return the intermediate CRC value, provided the CRC module is configured.

Address: 4003_2000h base + 0h offset = 4003_2000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HU								HL								LU								LL							
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_DATA field descriptions

Field	Description
31–24 HU	CRC High Upper Byte In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
23–16 HL	CRC High Lower Byte In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
15–8 LU	CRC Low Upper Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.
LL	CRC Low Lower Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.

22.2.2 CRC Polynomial register (CRC_GPOLY)

This register contains the value of the polynomial for the CRC calculation. The HIGH field contains the upper 16 bits of the CRC polynomial, which are used only in 32-bit CRC mode. Writes to the HIGH field are ignored in 16-bit CRC mode. The LOW field contains the lower 16 bits of the CRC polynomial, which are used in both 16- and 32-bit CRC modes.

Address: 4003_2000h base + 4h offset = 4003_2004h

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

CRC_GPOLY field descriptions

Field	Description
31–16 HIGH	High Polynominal Half-word Writable and readable in 32-bit CRC mode (CTRL[TCRC] is 1). This field is not writable in 16-bit CRC mode (CTRL[TCRC] is 0).
LOW	Low Polynominal Half-word Writable and readable in both 32-bit and 16-bit CRC modes.

22.2.3 CRC Control register (CRC_CTRL)

This register controls the configuration and working of the CRC module. Appropriate bits must be set before starting a new CRC calculation. A new CRC calculation is initialized by asserting CTRL[WAS] and then writing the seed into the CRC data register.

Address: 4003_2000h base + 8h offset = 4003_2008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TOT		TOTR		0	FXOR	WAS	TCRC	0							
W					▨				▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRC_CTRL field descriptions

Field	Description
31–30 TOT	Type Of Transpose For Writes Defines the transpose configuration of the data written to the CRC data register. See the description of the transpose feature for the available transpose options. 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed. 10 Both bits in bytes and bytes are transposed. 11 Only bytes are transposed; no bits in a byte are transposed.
29–28 TOTR	Type Of Transpose For Read Identifies the transpose configuration of the value read from the CRC Data register. See the description of the transpose feature for the available transpose options. 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed. 10 Both bits in bytes and bytes are transposed. 11 Only bytes are transposed; no bits in a byte are transposed.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FXOR	Complement Read Of CRC Data Register Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables on the fly complementing of read data. 0 No XOR on reading. 1 Invert or complement the read value of the CRC Data register.
25 WAS	Write CRC Data Register As Seed When asserted, a value written to the CRC data register is considered a seed value. When deasserted, a value written to the CRC data register is taken as data for CRC computation. 0 Writes to the CRC data register are data values. 1 Writes to the CRC data register are seed values.
24 TCRC	Width of CRC protocol. 0 16-bit CRC protocol. 1 32-bit CRC protocol.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.3 Functional description

22.3.1 CRC initialization/reinitialization

To enable the CRC calculation, the user must program CRC_CTRL[WAS], CRC_GPOLY, necessary parameters for transposition and CRC result inversion in the applicable registers. Asserting CRC_CTRL[WAS] enables the programming of the seed value into the CRC_DATA register.

After a completed CRC calculation, reasserting CRC_CTRL[WAS] and programming a seed, whether the value is new or a previously used seed value, reinitialize the CRC module for a new CRC computation. All other parameters must be set before programming the seed value and subsequent data values.

22.3.2 CRC calculations

In 16-bit and 32-bit CRC modes, data values can be programmed 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous. Noncontiguous bytes can lead to an incorrect CRC computation.

22.3.2.1 16-bit CRC

To compute a 16-bit CRC:

1. Clear CRC_CTRL[TCRC] to enable 16-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 16-bit polynomial to the CRC_GPOLY[LOW] field. The CRC_GPOLY[HIGH] field is not usable in 16-bit CRC mode.
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to CRC_DATA[LU:LL]. CRC_DATA[HU:HL] are not used.
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[LU:LL].

Transpose and complement operations are performed on the fly while reading or writing values. See Transpose feature and CRC result complement for details.

22.3.2.2 32-bit CRC

To compute a 32-bit CRC:

1. Set CRC_CTRL[TCRC] to enable 32-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 32-bit polynomial to CRC_GPOLY[HIGH:LOW].
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to CRC_DATA[HU:HL:LU:LL].
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[HU:HL:LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[HU:HL:LU:LL]. The CRC is calculated bytewise, and two clocks are required to complete one CRC calculation.

Transpose and complement operations are performed on the fly while reading or writing values. See Transpose feature and CRC result complement for details.

22.3.3 Transpose feature

By default, the transpose feature is not enabled. However, some CRC standards require the input data and/or the final checksum to be transposed. The user software has the option to configure each transpose operation separately, as desired by the CRC standard. The data is transposed on the fly while being read or written.

Some protocols use little endian format for the data stream to calculate a CRC. In this case, the transpose feature usefully flips the bits. This transpose option is one of the types supported by the CRC module.

22.3.3.1 Types of transpose

The CRC module provides several types of transpose functions to flip the bits and/or bytes, for both writing input data and reading the CRC result, separately using the CTRL[TOT] or CTRL[TOTR] fields, according to the CRC calculation being used.

The following types of transpose functions are available for writing to and reading from the CRC data register:

1. CTRL[TOT] or CTRL[TOTR] is 00.

No transposition occurs.

2. CTRL[TOT] or CTRL[TOTR] is 01

   Bits in a byte are transposed, while bytes are not transposed.

   reg[31:0] becomes {reg[24:31], reg[16:23], reg[8:15], reg[0:7]}

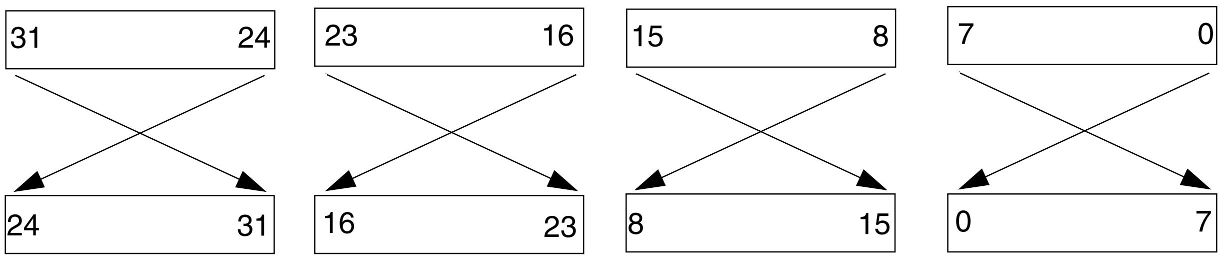


Figure 22-5. Transpose type 01

3. CTRL[TOT] or CTRL[TOTR] is 10.

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes = {reg[0:7], reg[8:15],reg[16:23], reg[24:31]}

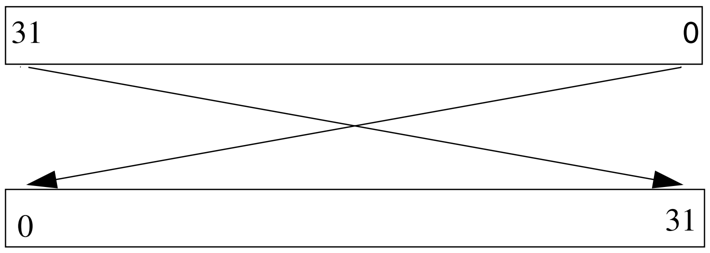


Figure 22-6. Transpose type 10

4. CTRL[TOT] or CTRL[TOTR] is 11.

   Bytes are transposed, but bits are not transposed.

   reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}

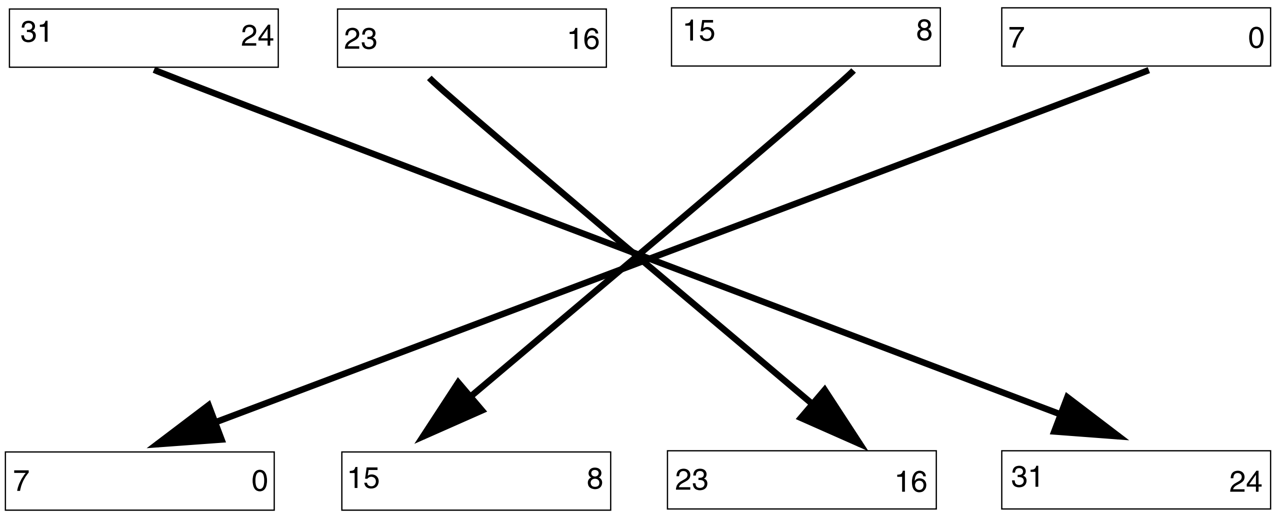


Figure 22-7. Transpose type 11

NOTE

For 8-bit and 16-bit write accesses to the CRC data register, the data is transposed with zeros on the unused byte or bytes (taking 32 bits as a whole), but the CRC is calculated on the valid byte(s) only. When reading the CRC data register for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in the CRC[HU:HL] fields. The user software must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

22.3.4 CRC result complement

When CTRL[FXOR] is set, the checksum is complemented. The CRC result complement function outputs the complement of the checksum value stored in the CRC data register every time the CRC data register is read. When CTRL[FXOR] is cleared, reading the CRC data register accesses the raw checksum value.

Chapter 23
Interrupt (IRQ)

23.1 Introduction

The external interrupt (IRQ) module provides a maskable interrupt input.

23.2 Features

Features of the IRQ module include:

- A dedicated external interrupt pin IRQ
- IRQ Interrupt Control bits
- Programmable edge-only or edge and level interrupt sensitivity
- Automatic interrupt acknowledge
- Internal pullup device

A low level applied to the external interrupt request (IRQ) pin can latch a CPU interrupt request. The following figure shows the structure of the IRQ module:

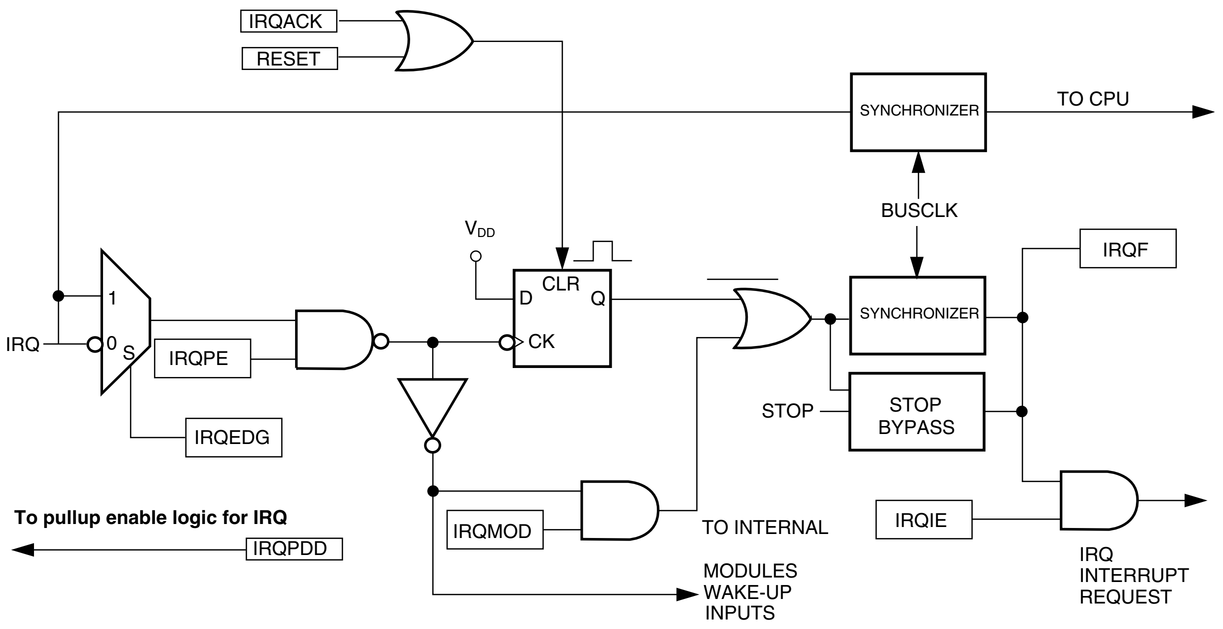


Figure 23-1. IRQ module block diagram

External interrupts are managed by the IRQSC status and control register. When the IRQ function is enabled, synchronous logic monitors the pin for edge-only or edge-and-level events. When the MCU is in Stop mode and system clocks are shut down, a separate asynchronous path is used so that the IRQ, if enabled, can wake the MCU.

23.2.1 Pin configuration options

The IRQ Pin Enable (IRQSC[IRQPE]) control field must be 1 for the IRQ pin to act as the IRQ input. The user can choose the polarity of edges or levels detected (IRQEDG), whether the pin detects edges-only or edges and levels (IRQMOD), or whether an event causes an interrupt or only sets the IRQSC[IRQF] flag, which can be polled by software.

When enabled, the IRQ pin defaults to use an internal pullup device (IRQSC[IRQPDD] = 0). If the user uses an external pullup or pulldown, the IRQSC[IRQPDD] can be written to a 1 to turn off the internal device.

BIH and BIL instructions may be used to detect the level on the IRQ pin when it is configured to act as the IRQ input.

Note

This pin does not contain a clamp diode to V_{DD} and must not be driven above V_{DD}. The voltage measured on the internally pullup IRQ pin may be as low as $V_{DD} – 0.7$ V. The internal gates connected to this pin are pulled all the way to V_{DD}.

When enabling the IRQ pin for use, IRQSC[IRQF] will be set, and must be cleared prior to enabling the interrupt. When configuring the pin for falling edge and level sensitivity in a 3 V system, it is necessary to wait at least cycles between clearing the flag and enabling the interrupt.

23.2.2 Edge and level sensitivity

The IRQSC[IRQMOD] control field reconfigures the detection logic so that it can detect edge events and pin levels. In this detection mode, IRQSC[IRQF] status flag is set when an edge is detected, if the IRQ pin changes from the deasserted to the asserted level, but the flag is continuously set and cannot be cleared as long as the IRQ pin remains at the asserted level.

23.3 Interrupt pin request register

IRQ memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_1000	Interrupt Pin Request Status and Control Register (IRQ_SC)	8	R/W	00h	23.3.1/299

23.3.1 Interrupt Pin Request Status and Control Register (IRQ_SC)

This direct page register includes status and control bits, which are used to configure the IRQ function, report status, and acknowledge IRQ events.

Address: 4003_1000h base + 0h offset = 4003_1000h

Bit	7	6	5	4	3	2	1	0
Read	0	IRQPDD	IRQEDG	IRQPE	IRQF	0	IRQIE	IRQMOD
Write	▨				▨	IRQACK		
Reset	0	0	0	0	0	0	0	0

IRQ_SC field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 IRQPDD	Interrupt Request (IRQ) Pull Device Disable

Table continues on the next page...

IRQ_SC field descriptions (continued)

Field	Description
	This read/write control bit is used to disable the internal pullup device when the IRQ pin is enabled (IRQPE = 1) allowing for an external device to be used. 0 IRQ pull device enabled if IRQPE = 1. 1 IRQ pull device disabled if IRQPE = 1.
5 IRQEDG	Interrupt Request (IRQ) Edge Select This read/write control field is used to select the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control field determines whether the IRQ pin is sensitive to both edges and levels or only edges. When the IRQ pin is enabled as the IRQ input and is configured to detect rising edges, the optional pullup resistor is disabled. 0 IRQ is falling-edge or falling-edge/low-level sensitive. 1 IRQ is rising-edge or rising-edge/high-level sensitive.
4 IRQPE	IRQ Pin Enable This read/write control field enables the IRQ pin function. When this field is set, the IRQ pin can be used as an interrupt request. 0 IRQ pin function is disabled. 1 IRQ pin function is enabled.
3 IRQF	IRQ Flag This read-only status field indicates when an interrupt request event has occurred. 0 No IRQ request 1 IRQ event is detected.
2 IRQACK	IRQ Acknowledge This write-only field is used to acknowledge interrupt request events (write 1 to clear IRQF). Writing 0 has no meaning or effect. Reads always return 0. If edge-and-level detection is selected (IRQMOD = 1), IRQF cannot be cleared while the IRQ pin remains at its asserted level.
1 IRQIE	IRQ Interrupt Enable This read/write control field determines whether IRQ events generate an interrupt request. 0 Interrupt request when IRQF set is disabled (use polling). 1 Interrupt requested whenever IRQF = 1.
0 IRQMOD	IRQ Detection Mode This read/write control field selects either edge-only detection or edge-and-level detection. 0 IRQ event is detected only on falling/rising edges. 1 IRQ event is detected on falling/rising edges and low/high levels.

Chapter 24
Analog-to-digital converter (ADC)

24.1 Introduction

The 12-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

24.1.1 Features

Features of the ADC module include:

- Linear Successive Approximation algorithm with 8-, 10-, or 12-bit resolution
- Up to 12 external analog inputs, external pin inputs, and 5 internal analog inputs including internal bandgap, temperature sensor, and references
- Output formatted in 8-, 10-, or 12-bit right-justified unsigned format
- Single or Continuous Conversion (automatic return to idle after single conversion)
- Support up to eight result FIFO with selectable FIFO depth
- Configurable sample time and conversion speed/power
- Conversion complete flag and interrupt
- Input clock selectable from up to four sources
- Operation in Wait or Stop modes for lower noise operation
- Asynchronous clock source for lower noise operation
- Selectable asynchronous hardware conversion trigger
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value

24.1.2 Block Diagram

This figure provides a block diagram of the ADC module.

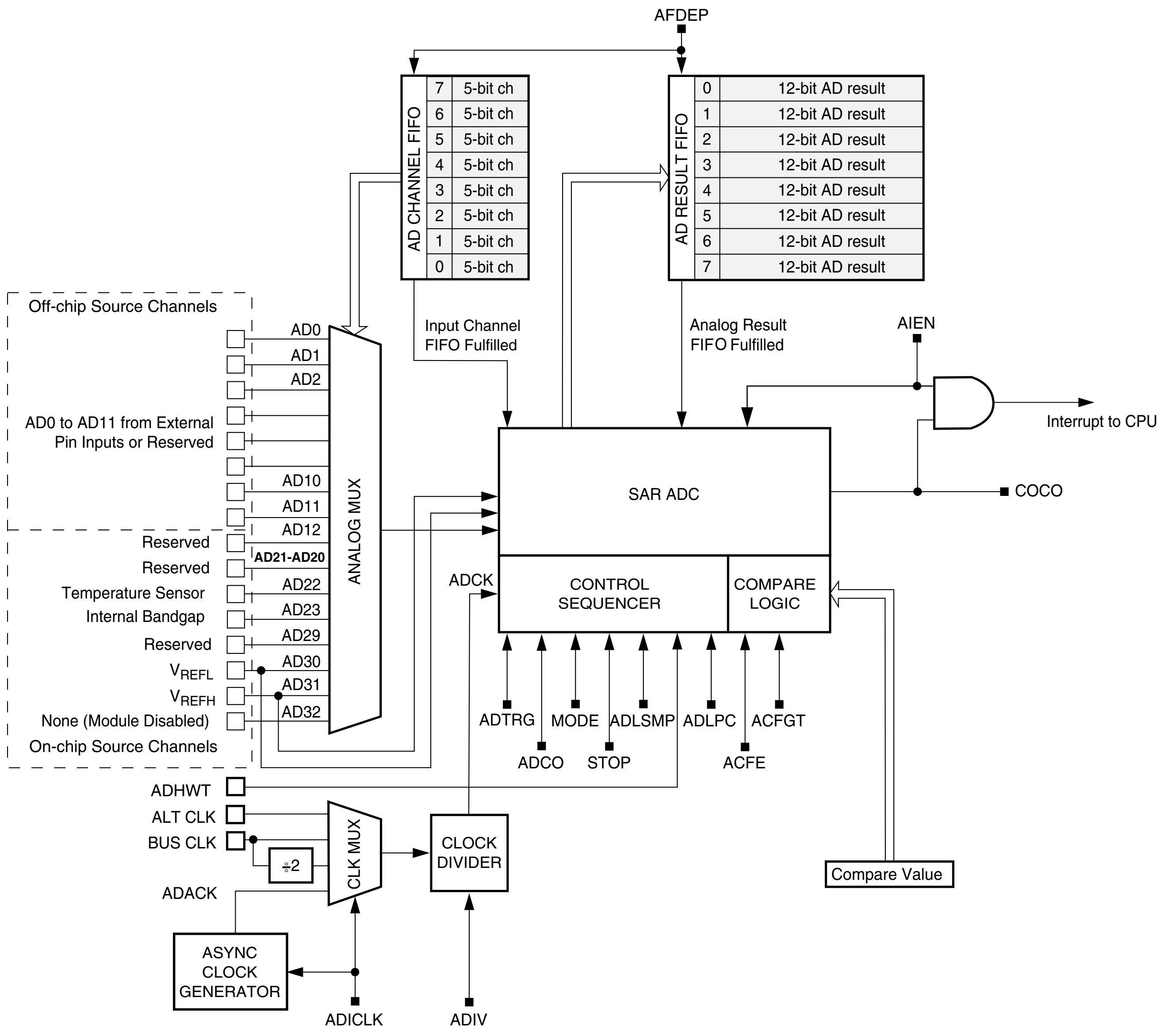


Figure 24-1. ADC Block Diagram

24.2 External Signal Description

The ADC module supports up to 24 separate analog inputs. It also requires four supply/reference/ground connections.

Table 24-1. Signal Properties

Name	Function
AD23–AD0	Analog Channel inputs
V_{REFH}	High reference voltage
V_{REFL}	Low reference voltage
V_{DDA}	Analog power supply
V_{SSA}	Analog ground

24.2.1 Analog Power (V_{DDA})

The ADC analog portion uses V_{DDA} as its power connection. In some packages, V_{DDA} is connected internally to V_{DD}. If externally available, connect the V_{DDA} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDA} for good results.

24.2.2 Analog Ground (V_{SSA})

The ADC analog portion uses V_{SSA} as its ground connection. In some packages, V_{SSA} is connected internally to V_{SS}. If externally available, connect the V_{SSA} pin to the same voltage potential as V_{SS}.

24.2.3 Voltage Reference High (V_{REFH})

V_{REFH} is the high reference voltage for the converter. In some packages, V_{REFH} is connected internally to V_{DDA}. If externally available, V_{REFH} may be connected to the same potential as V_{DDA} or may be driven by an external source between the minimum V_{DDA} specified in the data sheet and the V_{DDA} potential (V_{REFH} must never exceed V_{DDA}).

24.2.4 Voltage Reference Low (V_{REFL})

V_{REFL} is the low-reference voltage for the converter. In some packages, V_{REFL} is connected internally to V_{SSA}. If externally available, connect the V_{REFL} pin to the same voltage potential as V_{SSA}.

24.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 24 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

24.3 ADC Control Registers

ADC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_B000	Status and Control Register 1 (ADC_SC1)	32	R/W	0000_001Fh	24.3.1/304
4003_B004	Status and Control Register 2 (ADC_SC2)	32	R/W	0000_0008h	24.3.2/307
4003_B008	Status and Control Register 3 (ADC_SC3)	32	R/W	0000_0000h	24.3.3/309
4003_B00C	Status and Control Register 4 (ADC_SC4)	32	R/W	0000_0000h	24.3.4/310
4003_B010	Conversion Result Register (ADC_R)	32	R	0000_0000h	24.3.5/311
4003_B014	Compare Value Register (ADC_CV)	32	R/W	0000_0000h	24.3.6/312
4003_B018	Pin Control 1 Register (ADC_APCTL1)	32	R/W	0000_0000h	24.3.7/313
4003_B01C	Status and Control Register 5 (ADC_SC5)	32	R/W	0000_0000h	24.3.8/313

24.3.1 Status and Control Register 1 (ADC_SC1)

This section describes the function of the ADC status and control register (ADC_SC1). Writing ADC_SC1 aborts the current conversion and initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

When FIFO is enabled, the analog input channel FIFO is written via ADCH. The analog input channel queue must be written to ADCH continuously. The resulting FIFO follows the order in which the analog input channel is written. The ADC will start conversion when the input channel FIFO is fulfilled at the depth indicated by the ADC_SC4[AFDEP]. Any write 0x1F to these bits will reset the FIFO and stop the conversion if it is active.

Address: 4003_B000h base + 0h offset = 4003_B000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								COCO	AIEN	ADCO	ADCH				
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

ADC_SC1 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 COCO	Conversion Complete Flag Conversion Complete Flag. The COCO flag is a read-only bit set each time a conversion is completed when the compare function is disabled (ADC_SC2[ACFE] = 0). When the compare function is enabled (ADC_SC2[ACFE] = 1), the COCO flag is set upon completion of a conversion only if the compare result is true. When the FIFO function is enabled (ADC_SC4[AFDEP] > 0), the COCO flag is set upon completion of the set of FIFO conversion. This bit is cleared when ADC_SC1 is written or when ADC_R is read. 0 Conversion not completed. 1 Conversion completed.
6 AIEN	Interrupt Enable AIEN enables conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled. 1 Conversion complete interrupt enabled.
5 ADCO	Continuous Conversion Enable

Table continues on the next page...

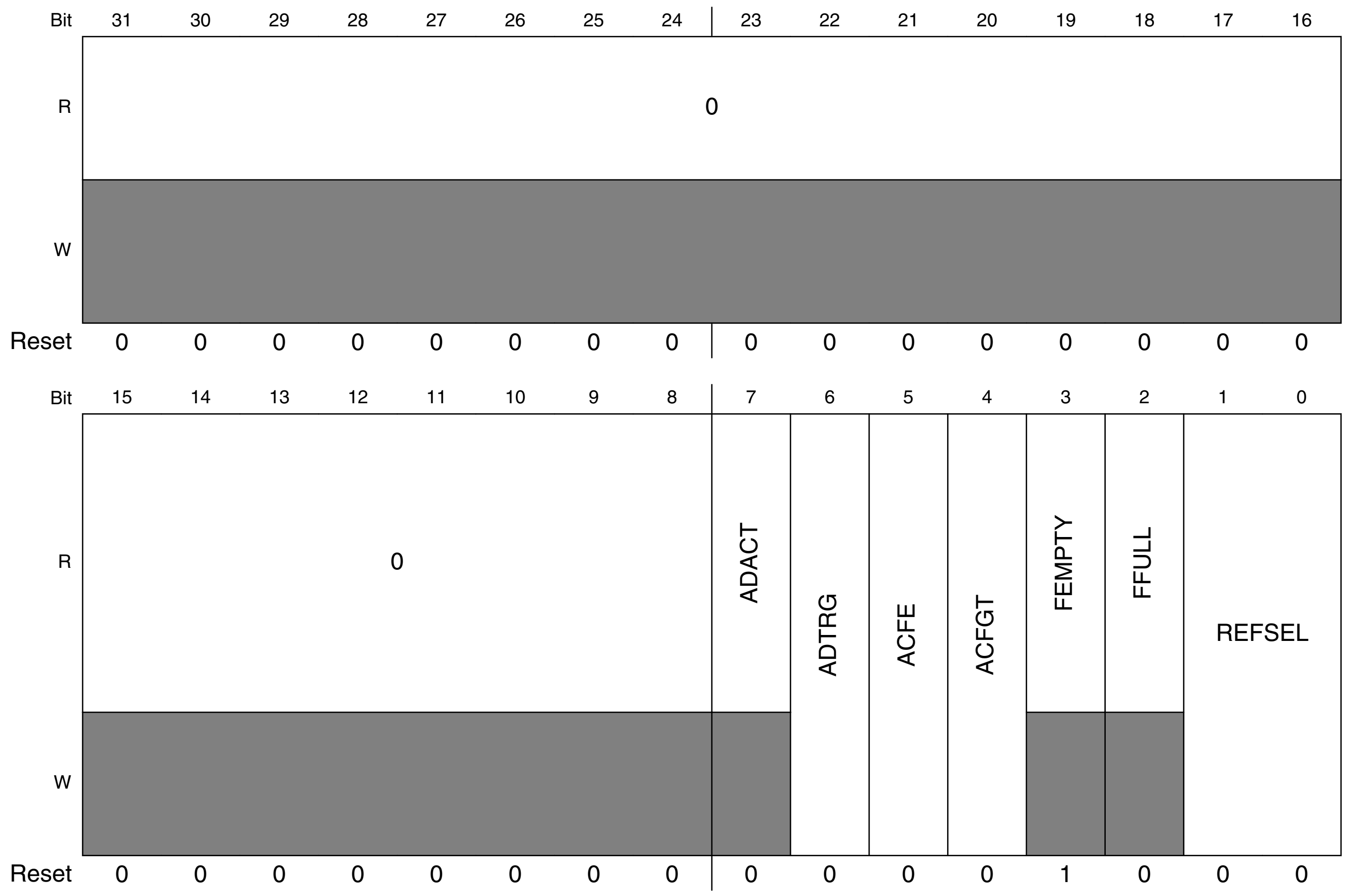
ADC_SC1 field descriptions (continued)

Field	Description
	ADCO enables continuous conversions. 0 One conversion following a write to the ADC_SC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. When the FIFO function is enabled (AFDEP > 0), a set of conversion are triggered when ADC_SC2[ADTRG]=0 or both ADC_SC2[ADTRG]=1 and ADC_SC4[HTRGME]=1. 1 Continuous conversions are initiated following a write to ADC_SC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected. When the FIFO function is enabled (AFDEP > 0), a set of conversions are loop triggered.
ADCH	Input Channel Select The ADCH bits form a 5-bit field that selects one of the input channels. 00000-01011 AD0-AD11 01100-10011 V_{SS} 10100-10101 Reserved 10110 Temperature Sensor 10111 Bandgap 11000-11100 Reserved 11101 V_{REFH} 11110 V_{REFL} 11111 Module disabled **NOTE:** Reset FIFO in FIFO mode.

24.3.2 Status and Control Register 2 (ADC_SC2)

The ADC_SC2 register controls the compare function, conversion trigger, and conversion active of the ADC module.

Address: 4003_B000h base + 4h offset = 4003_B004h



Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R 0
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R 0 ADACT ADTRG ACFE ACFGT FEMPTY FFULL REFSEL
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

ADC_SC2 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADACT	Conversion Active Indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress. 1 Conversion in progress.
6 ADTRG	Conversion Trigger Select Selects the type of trigger used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write

Table continues on the next page...

ADC_SC2 field descriptions (continued)

Field	Description
	to ADC_SC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input. 0 Software trigger selected. 1 Hardware trigger selected.
5 ACFE	Compare Function Enable Enables the compare function. 0 Compare function disabled. 1 Compare function enabled.
4 ACFGT	Compare Function Greater Than Enable Configures the compare function to trigger when the result of the conversion of the input being monitored is greater than or equal to the compare value. The compare function defaults to triggering when the result of the compare of the input being monitored is less than the compare value. 0 Compare triggers when input is less than compare level. 1 Compare triggers when input is greater than or equal to compare level.
3 FEMPTY	Result FIFO empty 0 Indicates that ADC result FIFO have at least one valid new data. 1 Indicates that ADC result FIFO have no valid new data.
2 FFULL	Result FIFO full 0 Indicates that ADC result FIFO is not full and next conversion data still can be stored into FIFO. 1 Indicates that ADC result FIFO is full and next conversion will override old data in case of no read action.
REFSEL	Voltage Reference Selection Selects the voltage reference source used for conversions. 00 Default voltage reference pin pair (V_{REFH}/V_{REFL}). 01 Analog supply pin pair (V_{DDA}/V_{SSA}). 10 Reserved. 11 Reserved - Selects default voltage reference (V_{REFH}/V_{REFL}) pin pair.

24.3.3 Status and Control Register 3 (ADC_SC3)

ADC_SC3 selects the mode of operation, clock source, clock divide, and configure for low power or long sample time.

Address: 4003_B000h base + 8h offset = 4003_B008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								ADLPC	ADIV		ADLSMP	MODE		ADICLK	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_SC3 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADLPC	Low-Power Configuration ADLPC controls the speed and power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 High speed configuration. 1 Low power configuration:The power is reduced at the expense of maximum clock speed.
6–5 ADIV	Clock Divide Select ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. 00 Divide ration = 1, and clock rate = Input clock. 01 Divide ration = 2, and clock rate = Input clock ÷ 2. 10 Divide ration = 3, and clock rate = Input clock ÷ 4. 11 Divide ration = 4, and clock rate = Input clock ÷ 8.
4 ADLSMP	Long Sample Time Configuration ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time. 1 Long sample time.

Table continues on the next page...

ADC_SC3 field descriptions (continued)

Field	Description
3–2 MODE	Conversion Mode Selection MODE bits are used to select between 12-, 10-, or 8-bit operation. 00 8-bit conversion (N = 8) 01 10-bit conversion (N = 10) 10 12-bit conversion (N = 12) 11 Reserved
ADICLK	Input Clock Select ADICLK bits select the input clock source to generate the internal clock ADCK. 00 Bus clock 01 Bus clock divided by 2 10 Alternate clock (ALTCLK) 11 Asynchronous clock (ADACK)

24.3.4 Status and Control Register 4 (ADC_SC4)

This register controls the FIFO scan mode, FIFO compare function and FIFO depth selection of the ADC module.

Address: 4003_B000h base + Ch offset = 4003_B00Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							HTRGME	0	ASCANE	ACFSEL	0		AFDEP		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_SC4 field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 HTRGME	Hardware Trigger Multiple Conversion Enable This field enables hardware trigger multiple conversion.

Table continues on the next page...

ADC_SC4 field descriptions (continued)

Field	Description
	0 One hardware trigger pulse triggers one conversion. 1 One hardware trigger pulse triggers multiple conversions in fifo mode.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 ASCANE	FIFO Scan Mode Enable The FIFO always use the first dummied FIFO channels when it is enabled. When this bit is set and FIFO function is enabled, ADC will repeat using the first FIFO channel as the conversion channel until the result FIFO is fulfilled. In continuous mode (ADCO = 1), ADC will start next conversion with the same channel when COCO is set. 0 FIFO scan mode disabled. 1 FIFO scan mode enabled.
5 ACFSEL	Compare Function Selection Compare function select OR/AND when the FIFO function is enabled (AFDEP > 0). When this field is cleared, ADC will OR all of compare triggers and set COCO after at least one of compare trigger occurs. When this field is set, ADC will AND all of compare triggers and set COCO after all of compare triggers occur. 0 OR all of compare trigger. 1 AND all of compare trigger.
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
AFDEP	FIFO Depth FIFO Depth enables the FIFO function and sets the depth of FIFO. When AFDEP is cleared, the FIFO is disabled. When AFDEP is set to nonzero, the FIFO function is enabled and the depth is indicated by the AFDEP bits. The ADC_SC1[ADCH] and ADC_R must be accessed by FIFO mode when FIFO function is enabled. ADC starts conversion when the analog channel FIFO is upon the level indicated by AFDEP bits. The COCO bit is set when the set of conversions are completed and the result FIFO is upon the level indicated by AFDEP bits. 000 FIFO is disabled. 001 2-level FIFO is enabled. 010 3-level FIFO is enabled.. 011 4-level FIFO is enabled. 100 5-level FIFO is enabled. 101 6-level FIFO is enabled. 110 7-level FIFO is enabled. 111 8-level FIFO is enabled.

24.3.5 Conversion Result Register (ADC_R)

In 12-bit operation, ADC_R contains the 12 bits of the result of a 12-bit conversion.

In 10-bit mode, ADC_R contains the 10 bits of the result of a 10-bit conversion.

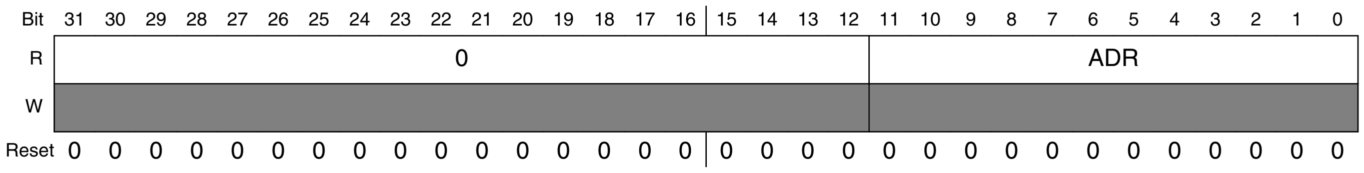
In 8-bit mode, ADC_R contains the 8 bits of the result of a 8-bit conversion.

ADC_R is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met.

When FIFO is enabled, the result FIFO is read via ADC_R. The ADC conversion completes when the input channel FIFO is fulfilled at the depth indicated by the AFDEP. The AD result FIFO can be read via ADC_R continuously by the order set in analog input channel ADCH.

If the MODE bits are changed, any data in ADC_R becomes invalid.

Address: 4003_B000h base + 10h offset = 4003_B010h



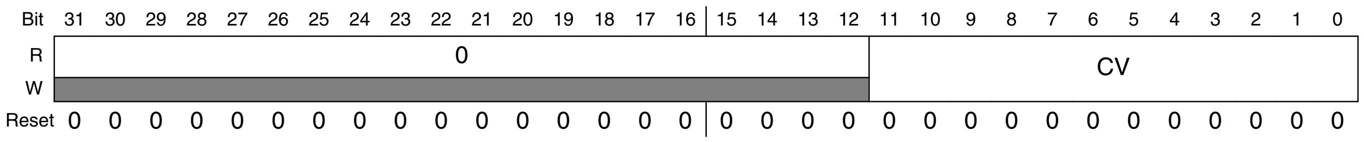
ADC_R field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ADR	Conversion Result

24.3.6 Compare Value Register (ADC_CV)

This register holds the compare value. Bits ADCV11:ADCV0 are compared to the 12 bits of the result following a conversion in 12-bit mode.

Address: 4003_B000h base + 14h offset = 4003_B014h



ADC_CV field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CV	Conversion Result[11:0]

24.3.7 Pin Control 1 Register (ADC_APCTL1)

The pin control registers disable the I/O port control of MCU pins used as analog inputs. APCTL1 is used to control the pins associated with channels 0-31 of the ADC module.

Address: 4003_B000h base + 18h offset = 4003_B018h

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

ADC_APCTL1 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ADPC	ADC Pin Control ADPCx controls the pin associated with channel ADx. 0 ADx pin I/O control enabled. 1 ADx pin I/O control disabled.

24.3.8 Status and Control Register 5 (ADC_SC5)

ADC_SC5 selects the hardware trigger mask.

Address: 4003_B000h base + 1Ch offset = 4003_B01Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0														HTRGMASKE	HTRGMASKSEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_SC5 field descriptions

Field	Description
31–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 HTRGMASKE	Hardware Trigger Mask Enable This field enables hardware trigger mask when HTRGMASKSEL is low. 0 Hardware trigger mask disable. 1 Hardware trigger mask enable and hardware trigger cannot trigger ADC conversion..
0 HTRGMASKSEL	Hardware Trigger Mask Mode Select This field selects hardware trigger mask mode. 0 Hardware trigger mask with HTRGMASKE. 1 Hardware trigger mask automatically when data fifo is not empty.

24.4 Functional description

The ADC module is disabled during reset or when the ADC_SC1[ADCH] bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle, the module is in its lowest power state.

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. In 12-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 12-bit digital result. In 10-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 10-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 8-bit digital result.

When the conversion is completed, the result is placed in the data registers (ADC_R). In 10-bit mode, the result is rounded to 10 bits and placed in the data registers (ADC_R). In 8-bit mode, the result is rounded to 8 bits and placed in ADC_R. The conversion complete flag (ADC_SC1[COCO]) is then set and an interrupt is generated if the conversion complete interrupt has been enabled (ADC_SC1[AIEN] = 1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of its compare registers. The compare function is enabled by setting the ADC_SC2[ACFE] bit and operates with any of the conversion modes and configurations.

24.4.1 Clock select and divide control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock to the converter (ADCK). The clock is selected from one of the following sources by means of the ADC_SC3[ADICLK] bits.

- The bus clock divided by 2: For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, that is, alternate clock which is OSC_OUT
- The asynchronous clock (ADACK): This clock is generated from a clock source within the ADC module. When selected as the clock source, this clock remains active while the MCU is in Wait or Stop mode and allows conversions in these modes for lower noise operation.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC does not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADC_SC3[ADIV] bits and can be divide-by 1, 2, 4, or 8.

24.4.2 Input select and pin control

The Pin Control register (ADC_APCTL1) disables the I/O port control of the pins used as analog inputs. When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

- The output buffer is forced to its high impedance state.
- The input buffer is disabled. A read of the I/O port returns a zero for any pin with its input buffer disabled.
- The pullup is disabled.

24.4.3 Hardware trigger

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADC_SC2[ADTRG] bit is set. This source is not available on all MCUs. See the module introduction for information on the ADHWT source specific to this MCU.

When ADHWT source is available and hardware trigger is enabled (ADC_SC2[ADTRG] = 1), a conversion is initiated on the rising edge of ADHWT. If a conversion is in progress when a rising edge occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

24.4.4 Conversion control

Conversions can be performed in 12-bit mode, 10-bit mode, or 8-bit mode as determined by the ADC_SC3[MODE] bits. Conversions can be initiated by a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, and an automatic compare of the conversion result to a software determined compare value.

24.4.4.1 Initiating conversions

A conversion initiates under the following conditions:

- A write to ADC_SC1 or a set of write to ADC_SC1 in FIFO mode (with ADCH bits not all 1s) if software triggered operation is selected.
- A hardware trigger (ADHWT) event if hardware triggered operation is selected.
- The transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADC_SC1 is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

24.4.4.2 Completing conversions

A conversion is completed when the result of the conversion is transferred into the data result register, ADC_R. This is indicated by the setting of ADC_SC1[COCO]. An interrupt is generated if ADC_SC1[AIEN] is high at the time that ADC_SC1[COCO] is set.

24.4.4.3 Aborting conversions

Any conversion in progress is aborted in the following cases:

- A write to ADC_SC1 occurs.
  - The current conversion will be aborted and a new conversion will be initiated, if ADC_SC1[ADCH] are not all 1s and ADC_SC4[AFDEP] are all 0s.
  - The current conversion and the rest of conversions will be aborted and no new conversion will be initialed, if ADC_SC4[AFDEP] are not all 0s.
  - A new conversion will be initiated when the FIFO is re-fulfilled upon the levels indicated by the ADC_SC4[AFDEP] bits).
- A write to ADC_SC2, ADC_SC3, ADC_SC4, ADC_CV occurs. This indicates a mode of operation change has occurred and the current and rest of conversions (when ADC_SC4[AFDEP] are not all 0s) are therefore invalid.
- The MCU is reset.
- The MCU enters Stop mode with ADACK not enabled.

When a conversion is aborted, the contents of the data register, ADC_R, are not altered. However, they continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset, ADC_R returns to their reset states.

24.4.4.4 Power control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source, the ADACK clock generator is also enabled.

Power consumption when active can be reduced by setting ADC_SC3[ADLPC]. This results in a lower maximum value for f_{ADCK} (see the data sheet).

24.4.4.5 Sample time and total conversion time

The total conversion time depends on the sample time (as determined by ADC_SC3[ADLSMP]), the MCU bus frequency, the conversion mode (8-bit, 10-bit or 12-bit), and the frequency of the conversion clock (f_{ADCK}). After the module becomes active, sampling of the input begins.ADC_SC3[ADLSMP] selects between short (3.5 ADCK cycles) and long (23.5 ADCK cycles) sample times. When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The result of the conversion is transferred to ADC_R upon completion of the conversion algorithm.

If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled (ADC_SC3[ADLSMP] = 0). If the bus frequency is less than 1/11th of the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when long sample is enabled (ADC_SC3[ADLSMP] = 1).

The maximum total conversion time for different conditions is summarized in the table below.

Table 24-11. Total conversion time vs. control conditions

Conversion type	ADICLK	ADLSMP	Max total conversion time
Single or first continuous 8-bit	0x, 10	0	20 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 μs + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	0	5 μs + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 μs + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	1	5 μs + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} > f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} > f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} > f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} > f_{ADCK}/11$	xx	1	40 ADCK cycles

The maximum total conversion time is determined by the selected clock source and the divide ratio. The clock source is selectable by the ADC_SC3[ADICLK] bits, and the divide ratio is specified by the ADC_SC3[ADIV] bits. For example, in 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 8 MHz, then the conversion time for a single conversion as given below:

$$\text{Conversion time} = \frac{23\,\text{ADCK Cyc}}{8\text{MHz}/1} + \frac{5\,\text{bus Cyc}}{8\text{MHz}} = 3.5\,\mu\text{s}$$

The number of bus cycles at 8 MHz is:

$$\text{Bus cycles} = 3.5\mu\text{s} \times 8\text{MHz} = 28$$

Note

> The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

24.4.5 Automatic compare function

The compare function can be configured to check for an upper or lower limit. After the input is sampled and converted, the result is added to the complement of the compare value (ADC_CV). When comparing to an upper limit (ADC_SC2[ACFGT] = 1), if the result is greater-than or equal-to the compare value, ADC_SC1[COCO] is set. When comparing to a lower limit (ADC_SC2[ACFGT] = 0), if the result is less than the compare value, ADC_SC1[COCO] is set. The value generated by the addition of the conversion result and the complement of the compare value is transferred to ADC_R.

On completion of a conversion while the compare function is enabled, if the compare condition is not true, ADC_SC1[COCO] is not set and no data is transferred to the result registers. An ADC interrupt is generated on the setting of ADC_SC1[COCO] if the ADC interrupt is enabled (ADC_SC1[AIEN] = 1).

On completion of all conversions while the compare function is enabled and FIFO enabled, if none of the compare conditions are not true when ADC_SC4[ACFSEL] is low or if not all of compare conditions are true when ADC_SC4[ACFSEL] is high, ADC_SC1[COCO] is not set. The compare data are transferred to the result registers regardless of compare condition true or false when FIFO enabled.

Note

> The compare function can monitor the voltage on a channel while the MCU is in Wait or Stop mode. The ADC interrupt wakes the MCU when the compare condition is met.

Note

The compare function can not work in continuous conversion mode when FIFO enabled.

24.4.6 FIFO operation

The ADC module supports FIFO operation to minimize the interrupts to CPU in order to reduce CPU loading in ADC interrupt service routines. This module contains two FIFOs to buffer analog input channels and analog results respectively.

The FIFO function is enabled when the ADC_SC4[AFDEP] bits are set non-zero. The FIFO depth is indicated by these bits. The FIFO supports up to eight level buffer.

The analog input channel FIFO is accessed by ADC_SC1[ADCH] bits, when FIFO function is enabled. The analog channel must be written to this FIFO in order. The ADC will not start the conversion if the channel FIFO is fulfilled below the level indicated by the ADC_SC4[AFDEP] bits, no matter whether software or hardware trigger is set. Read ADC_SC1[ADCH] will read the current active channel value. Write to ADC_SC1[ADCH] will re-fill channel FIFO to initial new conversion. It will abort current conversion and any other conversions that did not start. Write to the ADC_SC1 after all the conversions are completed or ADC is in idle state.

The result of the FIFO is accessed by ADC_R register, when FIFO function is enabled. The result must be read via these two registers by the same order of analog input channel FIFO to get the proper results. Don't read ADC_R until all of the conversions are completed in FIFO mode. The ADC_SC1[COCO] bit will be set only when all conversions indicated by the analog input channel FIFO complete whatever software or hardware trigger is set. An interrupt request will be submitted to CPU if the ADC_SC1[AIEN] is set when the FIFO conversion completes and the ADC_SC1[COCO] bit is set.

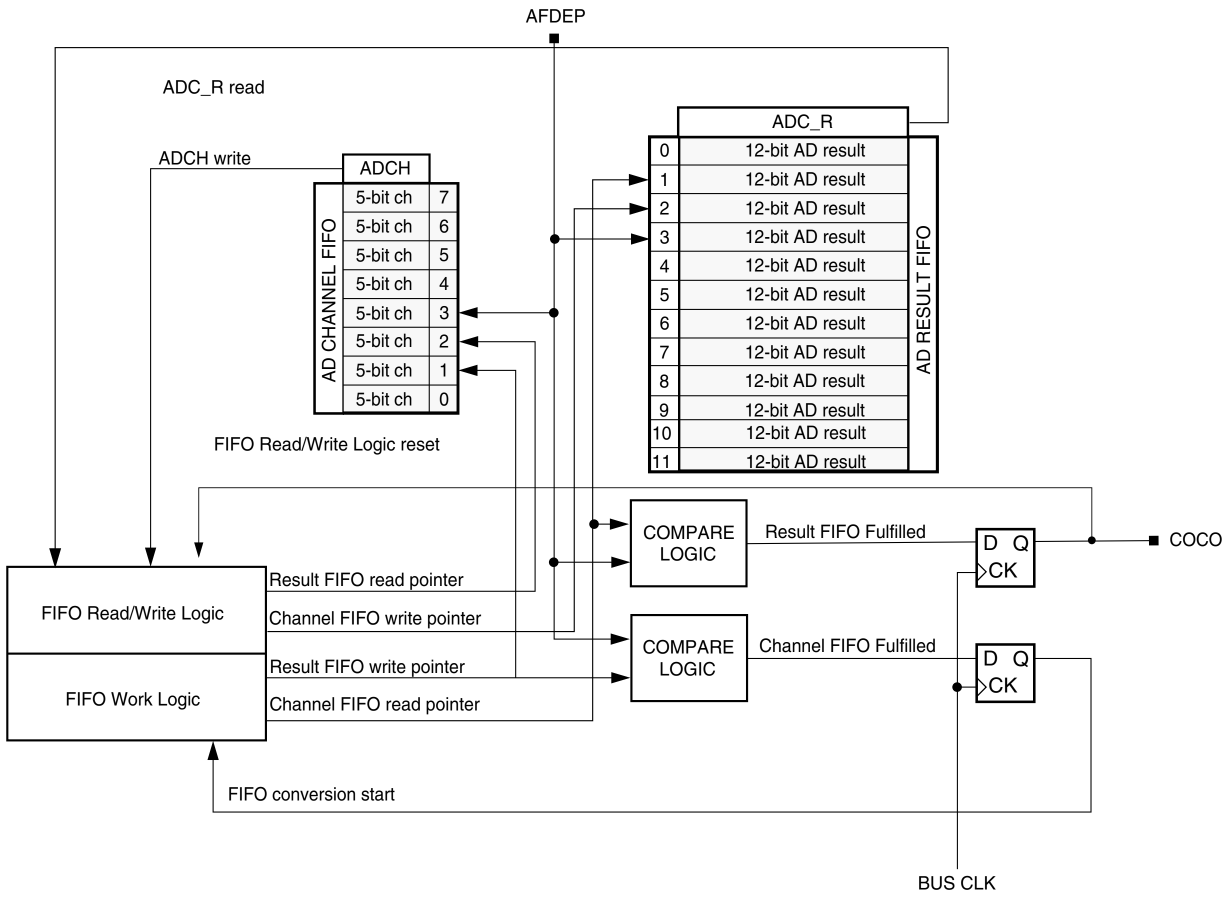


Figure 24-10. FADC FIFO structure

If software trigger is enabled, the next analog channel is fetched from analog input channel FIFO as soon as a conversion completes and its result is stored in the result FIFO. When all conversions set in the analog input channel FIFO completes, the ADC_SC1[COCO] bit is set and an interrupt request will be submitted to CPU if the ADC_SC1[AIEN] bit is set.

If single hardware trigger mode is enabled(ADC_SC2[ADTRG]= 1 and ADC_SC4[HTRGME]=0), the next analog is fetched from analog input channel FIFO only when this conversion completes, its result is stored in the result FIFO, and the next hardware trigger is fed to ADC module. If multi hardware tigger mode is enabled(ADC_SC2[ADTRG]=1 and ADC_SC4[HTRGME]=1), the next analog is fetched from analog input channel FIFO only when this conversion completes, its result is stored in the result FIFO, and next conversion will start without waiting for next hardware trigger. When all conversions set in the analog input channel FIFO completes, the ADC_SC1[COCO] bit is set and an interrupt request will be submitted to CPU if the ADC_SC1[AIEN] bit is set.

In single conversion in which ADC_SC1[ADCO] bit is clear, the ADC stops conversions when ADC_SC1[COCO] bit is set until the channel FIFO is fulfilled again or new hardware trigger occur.

The FIFO also provides scan mode to simplify the dummy work of input channel FIFO. When the ADC_SC4[ASCANE] bit is set in FIFO mode, the FIFO will always use the first dummied channel in spite of the value in the input channel FIFO. The ADC conversion start to work in FIFO mode as soon as the first channel is dummied. The following write operation to the input channel FIFO will cover the first channel element in this FIFO. In scan FIFO mode, the ADC_SC1[COCO] bit is set when the result FIFO is fulfilled according to the depth indicated by the ADC_SC4[AFDEP] bits.

In continuous conversion in which the ADC_SC1[ADCO] bit is set, the ADC starts next conversion immediately when all conversions are completed. ADC module will fetch the analog input channel from the beginning of analog input channel FIFO.

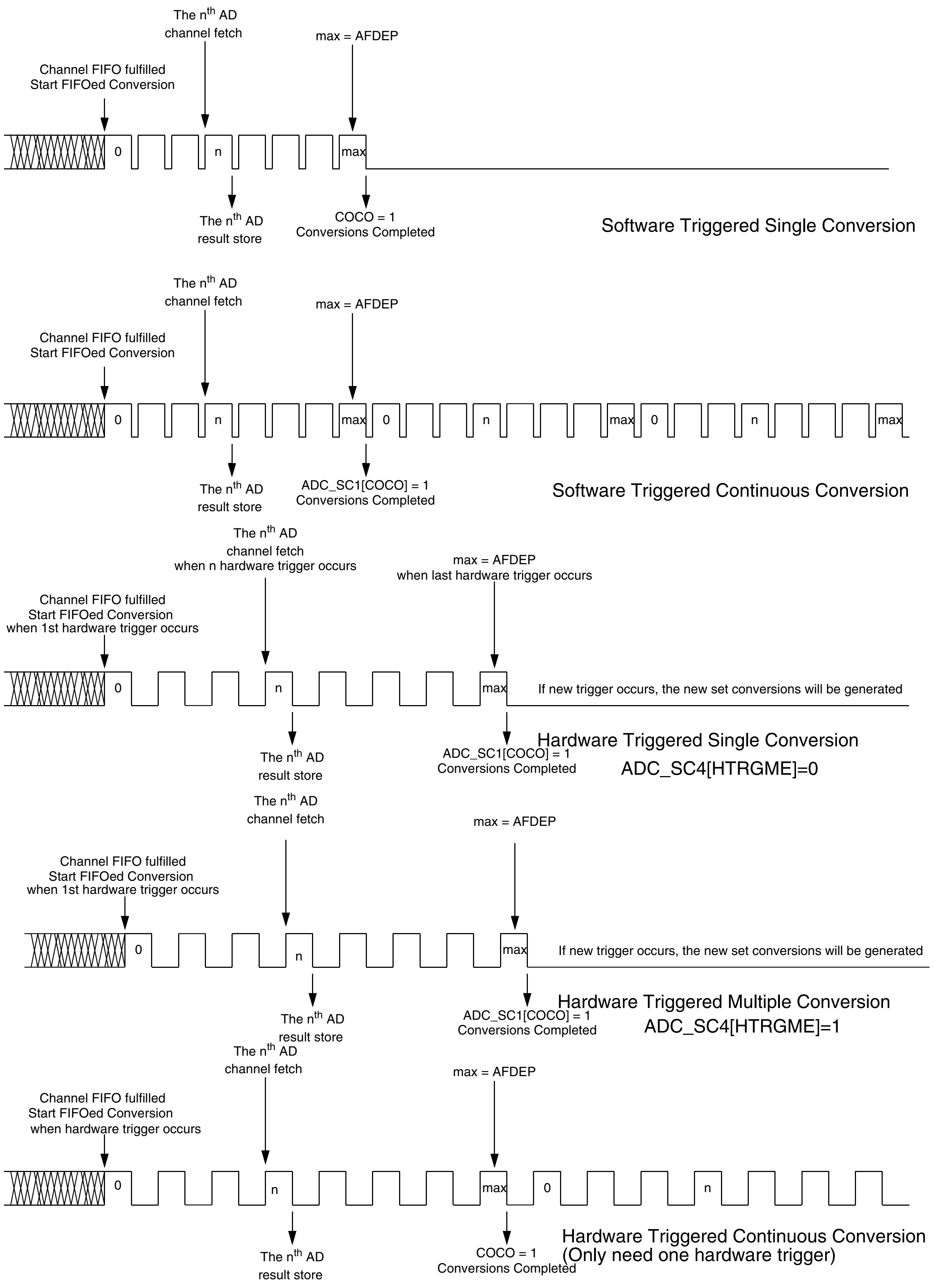

The nth AD channel fetch
max = AFDEP
Channel FIFO fulfilled Start FIFOed Conversion
0
n
max
The nth AD result store
COCO = 1 Conversions Completed
Software Triggered Single Conversion
The nth AD channel fetch
max = AFDEP
Channel FIFO fulfilled Start FIFOed Conversion
The nth AD result store
ADC_SC1[COCO] = 1 Conversions Completed
Software Triggered Continuous Conversion
The nth AD channel fetch when n hardware trigger occurs
max = AFDEP when last hardware trigger occurs
Channel FIFO fulfilled Start FIFOed Conversion when 1st hardware trigger occurs
If new trigger occurs, the new set conversions will be generated
The nth AD result store
ADC_SC1[COCO] = 1 Conversions Completed
Hardware Triggered Single Conversion
ADC_SC4[HTRGME]=0
The nth AD channel fetch
max = AFDEP
Channel FIFO fulfilled Start FIFOed Conversion when 1st hardware trigger occurs
If new trigger occurs, the new set conversions will be generated
The nth AD result store
ADC_SC1[COCO] = 1 Conversions Completed
Hardware Triggered Multiple Conversion
ADC_SC4[HTRGME]=1
The nth AD channel fetch
max = AFDEP
Channel FIFO fulfilled Start FIFOed Conversion when hardware trigger occurs
The nth AD result store
COCO = 1 Conversions Completed
Hardware Triggered Continuous Conversion (Only need one hardware trigger)

Figure 24-11. ADC FIFO conversion sequence

24.4.7 MCU wait mode operation

Wait mode is a low-power consumption standby mode from which recovery is fast because the clock sources remain active. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, ALTCLK and ADACK are available as conversion clock sources while in wait mode.

ADC_SC1[COCO] is set by a conversion complete event that generates an ADC interrupt to wake the MCU from wait mode if the ADC interrupt is enabled (ADC_SC1[AIEN] = 1).

24.4.8 MCU Stop mode operation

Stop mode is a low-power consumption standby mode during which most or all clock sources on the MCU are disabled.

24.4.8.1 Stop mode with ADACK disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a STOP instruction aborts the current conversion and places the ADC in its idle state. The contents of ADC_R are unaffected by Stop mode. After exiting from Stop mode, a software or hardware trigger is required to resume conversions.

24.4.8.2 Stop mode with ADACK enabled

If ADACK is selected as the conversion clock, the ADC continues operation during Stop mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during Stop mode. See the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters Stop mode, it continues until completion. Conversions can be initiated while the MCU is in Stop mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the ADC_SC1[COCO] and generates an ADC interrupt to wake the MCU from Stop mode if the ADC interrupt is enabled (ADC_SC1[AIEN] = 1). In fifo mode, ADC cannot complete the conversion operation fully or wake the MCU from Stop mode.

Note

The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, the data transfer blocking mechanism must be cleared when entering Stop and continuing ADC conversions.

24.5 Initialization information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. You can configure the module for 8-, 10-, or 12-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to ADC_SC3 register for information used in this example.

Note

Hexadecimal values prefixed by a 0x, binary values prefixed by a %, and decimal values have no preceding character.

24.5.1 ADC module initialization example

Before the ADC module can be used to complete conversions, it must be initialized. Given below is a method to initialize ADC module.

24.5.1.1 Initialization sequence

A typical initialization sequence is as follows:

1. Update the configuration register (ADC_SC3) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.
2. Update status and control register 2 (ADC_SC2) to select the hardware or software conversion trigger and compare function options, if enabled.

3. Update status and control register 1 (ADC_SC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

24.5.1.2 Pseudo-code example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from the bus clock divided by 1.

Example: 24.5.1.2.1 General ADC initialization routine

```
void ADC_init(void)
{
      /* The following code segment demonstrates how to initialize ADC by low-power mode, 
long
      sample time, bus frequency, software triggered from AD1 external pin without FIFO 
enabled
      */
      ADC_APCTL1 = ADC_APCTL1_ADPC1_MASK;
      ADC_SC3 = ADC_SC3_ADLPC_MASK | ADC_SC3_ADLSMP_MASK | ADC_SC3_MODE0_MASK;
      ADC_SC2 = 0x00;
      ADC_SC1 = ADC_SC1_AIEN_MASK | ADC_SC1_ADCH0_MASK;
}
```

24.5.2 ADC FIFO module initialization example

Before the ADC module can be used to start FIFOed conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Update the configuration register (ADC_SC3) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used to select sample time and low-power configuration.
2. Update the configuration register (ADC_SC4) to select the FIFO scan mode, FIFO compare function selection (OR or AND function) and FIFO depth.
3. Update status and control register 2 (ADC_SC2) to select the hardware or software conversion trigger, compare function options if enabled.
4. Update status and control register 1 (ADC_SC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

24.5.2.1 Pseudo-code example

In this example, the ADC module is set up with interrupts enabled to perform a single hardware triggered 10-bit 4-level-FIFO conversion at low power with a long sample time on input channels of 1, 3, 5, and 7. Here the internal ADCK clock is derived from the bus clock divided by 1.

Example: 24.5.2.1.1 FIFO ADC initialization routine

```
void ADC_init(void)
{
/* The following code segment demonstrates how to initialize ADC by low-power mode, long
sample time, bus frequency, hardware triggered from AD1, AD3, AD5, and AD7 external pins
with 4-level FIFO enabled */

ADC_APCTL1 = ADC_APCTL1_ADPC6_MASK | ADC_APCTL1_ADPC5_MASK | ADC_APCTL1_ADPC3_MASK |
ADC_APCTL1_ADPC1_MASK;

ADC_SC3 = ADC_SC3_ADLPC_MASK | ADC_SC3_ADLSMP_MASK | ADC_SC3_MODE1_MASK;


// setting hardware trigger
ADC_SC2 = ADC_SC2_ADTRG_MASK ;

//4-Level FIFO
ADC_SC4 = ADC_SC4_AFDEP1_MASK | ADC_SC4_AFDEP0_MASK;

// dummy the 1st channel
ADC_SC1 = ADC_SC1_ADCH0_MASK;

// dummy the 2nd channel
ADC_SC1 = ADC_SC1_ADCH1_MASK | ADC_SC1_ADCH0_MASK;

// dummy the 3rd channel
ADC_SC1 = ADC_SC1_ADCH2_MASK | ADC_SC1_ADCH0_MASK;

// dummy the 4th channel and ADC starts conversion
ADC_SC1 = ADC_SC1_AIEN_MASK | ADC_SC1_ADCH2_MASK | ADC_SC1_ADCH1_MASK | ADC_SC1_ADCH0_MASK;

}
```

Example: 24.5.2.1.2 FIFO ADC interrupt service routine

```
unsigned short buffer[4];
interrupt VectorNumber_Vadc void ADC_isr(void)
{
      /* The following code segment demonstrates read AD result FIFO */
      // read conversion result of channel 1 and COCO bit is cleared
      buffer[0] = ADC_R;
      // read conversion result of channel 3
      buffer[1] = ADC_R;
      // read conversion result of channel 5
      buffer[2] = ADC_R;
      // read conversion result of channel 7
      buffer[3] = ADC_R;
}
```

NOTE

ADC_R is 16-bit ADC result register, combined from ADC_RH and ADC_RL

24.6 Application information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

24.6.1 External pins and routing

The following sections discuss the external pins associated with the ADC module and how they are used for best results.

24.6.1.1 Analog supply pins

The ADC module has analog power and ground supplies (V_{DDA} and V_{SSA}) available as separate pins on some devices. V_{SSA} is shared on the same pin as the MCU digital V_{SS} on some devices. On other devices, V_{SSA} and V_{DDA} are shared with the MCU digital supply pins. In these cases, there are separate pads for the analog supplies bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

When available on a separate pin, both V_{DDA} and V_{SSA} must be connected to the same voltage potential as their corresponding MCU digital supply (V_{DD} and V_{SS}) and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

If separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the V_{SSA} pin. This should be the only ground connection between these supplies if possible. The V_{SSA} pin makes a good single point ground location.

24.6.1.2 Analog reference pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is V_{REFH}, which may be shared on the same pin as V_{DDA} on some devices. The low reference is V_{REFL}, which may be shared on the same pin as V_{SSA} on some devices.

When available on a separate pin, V_{REFH} may be connected to the same potential as V_{DDA}, or may be driven by an external source between the minimum V_{DDA} spec and the V_{DDA} potential (V_{REFH} must never exceed V_{DDA}). When available on a separate pin, V_{REFL} must be connected to the same voltage potential as V_{SSA}. V_{REFH} and V_{REFL} must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current causes a voltage drop that could result in conversion errors. Inductance in this path must be minimum (parasitic only).

24.6.1.3 Analog input pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer is in its high impedance state and the pullup is disabled. Also, the input buffer draws DC current when its input is not at V_{DD} or V_{SS}. Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 μF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to V_{SSA}.

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL}. If the input is equal to or exceeds V_{REFH}, the converter circuit converts the signal to 0xFFF (full scale 12-bit representation), 0x3FF (full scale 10-bit representation) or 0xFF (full scale 8-bit representation). If the input is equal to or less than V_{REFL}, the converter circuit converts it to 0x000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There is a brief current associated with V_{REFL} when the sampling capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADC_SC3[ADLSMP] is low, or 23.5 cycles when ADC_SC3[ADLSMP] is high.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins should not be transitioning during conversions.

24.6.2 Sources of error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

24.6.2.1 Sampling error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7 kΩ and input capacitance of approximately 5.5 pF, sampling to within 1/4 LSB (at 12-bit resolution) can be achieved within the minimum sample window (3.5 cycles at 8 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below 2 kΩ.

Higher source resistances or higher-accuracy sampling is possible by setting ADC_SC3[ADLSMP] (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

24.6.2.2 Pin leakage error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than $V_{DDA} / (2^N * I_{LEAK})$ for less than 1/4 LSB leakage error (N = 8 in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

24.6.2.3 Noise-induced errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 μF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 μF low-ESR capacitor from V_{DDA} to V_{SSA}.
- If inductive isolation is used from the primary supply, an additional 1 μF capacitor is placed from V_{DDA} to V_{SSA}.
- V_{SSA} (and V_{REFL}, if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or Stop mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
    - For software triggered conversions, immediately follow the write to ADC_SC1 with a stop instruction.
    - For Stop mode operation, select ADACK as the clock source. Operation in Stop reduces V_{DD} noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or Stop or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 μF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSA} (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

24.6.2.4 Code width and quantization error

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1\text{lsb} = (V_{\text{REFH}} - V_{\text{REFL}}) \,/\, 2^N$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2 lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is -1 lsb to 0 lsb and the code width of each step is 1 lsb.

24.6.2.5 Linearity errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system must be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1 lsb) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value that the absolute value of the running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

24.6.2.6 Code jitter, non-monotonicity, and missing codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter occurs when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate, between two codes, for a range of input voltages around the transition voltage. This range is normally around ±1/2 lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Noise-induced errors reduces this error.

Non-monotonicity is defined when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values that are never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

Chapter 25
Analog comparator (ACMP)

25.1 Introduction

The analog comparator module (ACMP) provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage (rail-to-rail operation).

The analog mux provides a circuit for selecting an analog input signal from four channels. One signal provided by the 6-bit DAC. The mux circuit is designed to operate across the full range of the supply voltage. The 6-bit DAC is 64-tap resistor ladder network which provides a selectable voltage reference for applications where voltage reference is needed. The 64-tap resistor ladder network divides the supply reference V_{in} into 64 voltage level. A 6-bit digital signal input selects output voltage level, which varies from V_{in} to $V_{in}/64$. V_{in} can be selected from two voltage sources.

25.1.1 Features

ACMP features include:

- Operational over the whole supply range of 2.7 V to 5.5 V
- On-chip 6-bit resolution DAC with selectable reference voltage from V_{DD} or internal bandgap
- Configurable hysteresis
- Selectable interrupt on rising edge, falling edge, or both rising or falling edges of comparator output
- Selectable inversion on comparator output
- Up to four selectable comparator inputs
- Operational in Stop mode

25.1.2 Modes of operation

This section defines the ACMP operation in Wait, Stop, and Background Debug modes.

25.1.2.1 Operation in Wait mode

The ACMP continues to operate in Wait mode, if enabled. The interrupt can wake the MCU if enabled.

25.1.2.2 Operation in Stop mode

The ACMP (including DAC and CMP) continues to operate in Stop mode if enabled. If ACMP_CS[ACIE] is set, a ACMP interrupt can be generated to wake the MCU up from Stop mode.

If the Stop is exited by an interrupt, the ACMP setting remains before entering the Stop mode. If Stop is exited with a reset, the ACMP goes into its reset.

The user must turn off the DAC if the output is not used as a reference input of ACMP to save power, because the DAC consumes additional power.

25.1.2.3 Operation in Debug mode

When the MCU is in Debug mode, the ACMP continues operating normally.

25.1.3 Block diagram

The block diagram of the ACMP module is shown in the following figure.

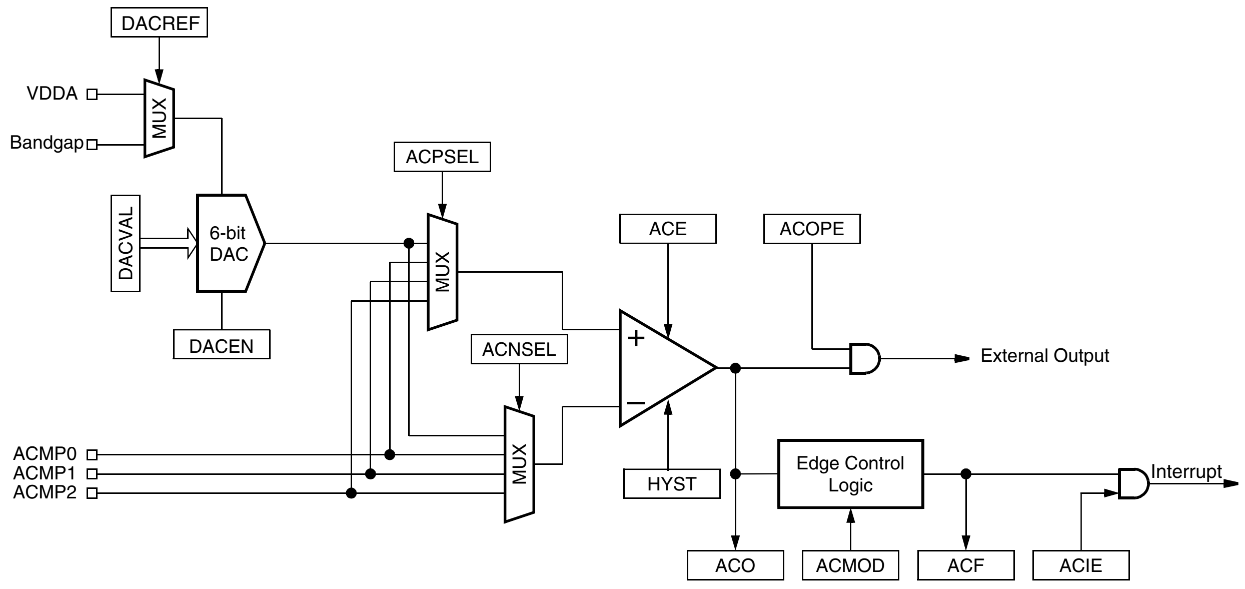


Figure 25-1. ACMP block diagram

25.2 External signal description

The output of ACMP can also be mapped to an external pin. When the output is mapped to an external pin, ACMP_CS[ACOPE] controls the pin to enable/disable the ACMP output function.

25.3 Memory map and register definition

ACMP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_3000	ACMP Control and Status Register (ACMP0_CS)	8	R/W	00h	25.3.1/338
4007_3001	ACMP Control Register 0 (ACMP0_C0)	8	R/W	00h	25.3.2/339
4007_3002	ACMP Control Register 1 (ACMP0_C1)	8	R/W	00h	25.3.3/339
4007_3003	ACMP Control Register 2 (ACMP0_C2)	8	R/W	00h	25.3.4/340
4007_4000	ACMP Control and Status Register (ACMP1_CS)	8	R/W	00h	25.3.1/338
4007_4001	ACMP Control Register 0 (ACMP1_C0)	8	R/W	00h	25.3.2/339
4007_4002	ACMP Control Register 1 (ACMP1_C1)	8	R/W	00h	25.3.3/339
4007_4003	ACMP Control Register 2 (ACMP1_C2)	8	R/W	00h	25.3.4/340

25.3.1 ACMP Control and Status Register (ACMPx_CS)

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read	ACE	HYST	ACF	ACIE	ACO	ACOPE	ACMOD	
Write					▨			
Reset	0	0	0	0	0	0	0	0

ACMPx_CS field descriptions

Field	Description
7 ACE	Analog Comparator Enable Enables the ACMP module. 0 The ACMP is disabled. 1 The ACMP is enabled.
6 HYST	Analog Comparator Hysterisis Selection Selects ACMP hysterisis. 0 20 mV. 1 30 mV.
5 ACF	ACMP Interrupt Flag Bit Synchronously set by hardware when ACMP output has a valid edge defined by ACMOD. The setting of this bit lags the ACMPO to bus clocks. Clear ACF bit by writing a 0 to this bit. Writing a 1 to this bit has no effect.
4 ACIE	ACMP Interrupt Enable Enables an ACMP CPU interrupt. 0 Disable the ACMP Interrupt. 1 Enable the ACMP Interrupt.
3 ACO	ACMP Output Reading ACO will return the current value of the analog comparator output. ACO is reset to a 0 and will read as a 0 when the ACMP is disabled (ACE = 0)
2 ACOPE	ACMP Output Pin Enable ACOPE enables the pad logic so that the output can be placed onto an external pin. 0 ACMP output cannot be placed onto external pin. 1 ACMP output can be placed onto external pin.
ACMOD	ACMP MOD Determines the sensitivity modes of the interrupt trigger. 00 ACMP interrupt on output falling edge. 01 ACMP interrupt on output rising edge. 10 ACMP interrupt on output falling edge. 11 ACMP interrupt on output falling or rising edge.

25.3.2 ACMP Control Register 0 (ACMPx_C0)

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read	0		ACPSEL		0		ACNSEL	
Write	▨				▨			
Reset	0	0	0	0	0	0	0	0

ACMPx_C0 field descriptions

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–4 ACPSEL	ACMP Positive Input Select 00 External reference 0 01 External reference 1 10 External reference 2 11 DAC output
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ACNSEL	ACMP Negative Input Select 00 External reference 0 01 External reference 1 10 External reference 2 11 DAC output

25.3.3 ACMP Control Register 1 (ACMPx_C1)

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read Write	DACEN	DACREF	DACVAL					
Reset	0	0	0	0	0	0	0	0

ACMPx_C1 field descriptions

Field	Description
7 DACEN	DAC Enable Enables the output of 6-bit DAC. 0 The DAC is disabled. 1 The DAC is enabled.
6 DACREF	DAC Reference Select

Table continues on the next page...

ACMPx_C1 field descriptions (continued)

Field	Description
	0 The DAC selects Bandgap as the reference. 1 The DAC selects V_{DDA} as the reference.
DACVAL	DAC Output Level Selection Selects the output voltage using the given formula: V_{output}= (V_{in}/64)x(DACVAL[5:0]+1) The V_{output} range is from V_{in}/64 to V_{in}, the step is V_{in}/64

25.3.4 ACMP Control Register 2 (ACMPx_C2)

Address: Base address + 3h offset

Bit	7	6	5	4	3	2	1	0
Read	0					ACIPE		
Write								
Reset	0	0	0	0	0	0	0	0

ACMPx_C2 field descriptions

Field	Description
7–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ACIPE	ACMP Input Pin Enable This 3-bit field controls if the corresponding ACMP external pin can be driven by an analog input. 0 The corresponding external analog input is not allowed. 1 The corresponding external analog input is allowed.

25.4 Functional description

The ACMP module is functionally composed of two parts: digital-to-analog (DAC) and comparator (CMP).

The DAC includes a 64-level DAC (digital to analog converter) and relevant control logic. DAC can select one of two reference inputs, V_{DD} or on-chip bandgap, as the DAC input V_{in} by setting ACMP_C1[DACREF]. After the DAC is enabled, it converts the data set in ACMP_C1[DACVAL] to a stepped analog output, which is fed into ACMP as an internal reference input. This stepped analog output is also mapped out of the module. The output voltage range is from V_{in}/64 to V_{in}. The step size is V_{in}/64.

The ACMP can achieve the analog comparison between positive input and negative input, and then give out a digital output and relevant interrupt. Both the positive and negative input of ACMP can be selected from the four common inputs: three external reference inputs and one internal reference input from the DAC output. The positive input of ACMP is selected by ACMP_C0[ACPSEL] and the negative input is selected by ACMP_C0[ACNSEL]. Any pair of the eight inputs can be compared by configuring the ACMPC0 with the appropriate value.

After the ACMP is enabled by setting ACMP_CS[ACE], the comparison result appears as a digital output. Whenever a valid edge defined in ACMP_CS[ACMOD] occurs, ACMP_CS[ACF] is asserted. If ACMP_CS[ACIE] is set, a ACMP CPU interrupt occurs. The valid edge is defined by ACMP_CS[ACMOD]. When ACMP_CS[ACMOD] = 00b or 10b, only the falling-edge on ACMP output is valid. When ACMP_CS[ACMOD] = 01b, only rising-edge on ACMP output is valid. When ACMP_CS[ACMOD] = 11b, both the rising-edge and falling-edge on the ACMP output are valid.

The ACMP output is synchronized by the bus clock to generate ACMP_CS[ACO] so that the CPU can read the comparison. In stop3 mode, if the output of ACMP is changed, ACMPO cannot be updated in time. The output can be synchronized and ACMP_CS[ACO] can be updated upon the waking up of the CPU because of the availability of the bus clock. ACMP_CS[ACO] changes following the comparison result, so it can serve as a tracking flag that continuously indicates the voltage delta on the inputs.

If a reference input external to the chip is selected as an input of ACMP, the corresponding ACMP_C2[ACIPE] bit must be set to enable the input from pad interface. If the output of the ACMP needs to be put onto the external pin, the ACMP_CS[ACOPE] bit must enable the ACMP pin function of pad logic.

25.5 Setup and operation of ACMP

The two parts of ACMP (DAC and CMP) can be set up and operated independently. But if the DAC works as an input of the CMP, the DAC must be configured before the ACMP is enabled.

Because the input-switching can cause problems on the ACMP inputs, the user should complete the input selection before enabling the ACMP and must not change the input selection setting when the ACMP is enabled to avoid unexpected output. Similarly, because the DAC experiences a setup delay after ACMP_C1[DACVAL] is changed, the user should complete the setting of ACMP_C1[DACVAL] before DAC is enabled.

25.6 Resets

During a reset the ACMP is configured in the default mode. Both CMP and DAC are disabled.

25.7 Interrupts

If the bus clock is available when a valid edge defined in ACMP_CS[ACMOD] occurs, the ACMP_CS[ACF] is asserted. If ACMP_CS[ACIE] is set, a ACMP interrupt event occurs. The ACMP_CS[ACF] bit remains asserted until the ACMP interrupt is cleared by software. When in stop mode, a valid edge on ACMP output generates an asynchronous interrupt that can wake the MCU from stop. The interrupt can be cleared by writing a 0 to the ACMP_CS[ACF] bit.

Chapter 26
FlexTimer Module (FTM)

26.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The FlexTimer module (FTM) is a two-to-eight channel timer that supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications. The FTM time reference is a 16-bit counter that can be used as an unsigned or signed counter.

26.1.1 FlexTimer philosophy

The FlexTimer is built upon a simple timer, the HCS08 Timer PWM Module – TPM, used for many years on Freescale's 8-bit microcontrollers. The FlexTimer extends the functionality to meet the demands of motor control, digital lighting solutions, and power conversion, while providing low cost and backwards compatibility with the TPM module.

Several key enhancements are made:

- Signed up counter
- Deadtime insertion hardware
- Fault control inputs
- Enhanced triggering functionality
- Initialization and polarity control

All of the features common with the TPM have fully backwards compatible register assignments. The FlexTimer can also use code on the same core platform without change to perform the same functions.

Motor control and power conversion features have been added through a dedicated set of registers and defaults turn off all new features. The new features, such as hardware deadtime insertion, polarity, fault control, and output forcing and masking, greatly reduce loading on the execution software and are usually each controlled by a group of registers.

FlexTimer input triggers can be from comparators, ADC, or other submodules to initiate timer functions automatically. These triggers can be linked in a variety of ways during integration of the sub modules so please note the options available for used FlexTimer configuration.

Several FlexTimers may be synchronized to provide a larger timer with their counters incrementing in unison, assuming the initialization, the input clocks, the initial and final counting values are the same in each FlexTimer.

All main user access registers are buffered to ease the load on the executing software. A number of trigger options exist to determine which registers are updated with this user defined data.

26.1.2 Features

The FTM features include:

- FTM source clock is selectable
    - Source clock can be the system clock, the fixed frequency clock, or an external clock
    - Fixed frequency clock is an additional clock input to allow the selection of an on chip clock source other than the system clock
    - Selecting external clock connects FTM clock to a chip level input pin therefore allowing to synchronize the FTM counter with an off chip clock source
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit counter
    - It can be a free-running counter or a counter with initial and final value
    - The counting can be up or up-down
- Each channel can be configured for input capture, output compare, or edge-aligned PWM mode
- In Input Capture mode:

    - The capture can occur on rising edges, falling edges or both edges
    - An input filter can be selected for some channels
- In Output Compare mode the output signal can be set, cleared, or toggled on match
- All channels can be configured for center-aligned PWM mode
- Each pair of channels can be combined to generate a PWM signal with independent control of both edges of PWM signal
- The FTM channels can operate as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs
- The deadtime insertion is available for each complementary pair
- Generation of match triggers
- Software control of PWM outputs
- Up to 4 fault inputs for global fault control
- The polarity of each channel is configurable
- The generation of an interrupt per channel
- The generation of an interrupt when the counter overflows
- The generation of an interrupt when the fault condition is detected
- Synchronized loading of write buffered FTM registers
- Write protection for critical registers
- Backwards compatible with TPM
- Testing of input captures for a stuck at zero and one conditions
- Dual edge capture for pulse and period width measurement

26.1.3 Modes of operation

When the MCU is in an active Debug mode, the FTM temporarily suspends all counting until the MCU returns to normal user operating mode. During Stop mode, all FTM input clocks are stopped, so the FTM is effectively disabled until clocks resume. During Wait mode, the FTM continues to operate normally. If the FTM does not need to produce a real time reference or provide the interrupt sources needed to wake the MCU from Wait mode, the power can then be saved by disabling FTM functions before entering Wait mode.

26.1.4 Block diagram

The FTM uses one input/output (I/O) pin per channel, CHn (FTM channel (n)) where n is the channel number (0–7).

The following figure shows the FTM structure. The central component of the FTM is the 16-bit counter with programmable initial and final values and its counting can be up or up-down.

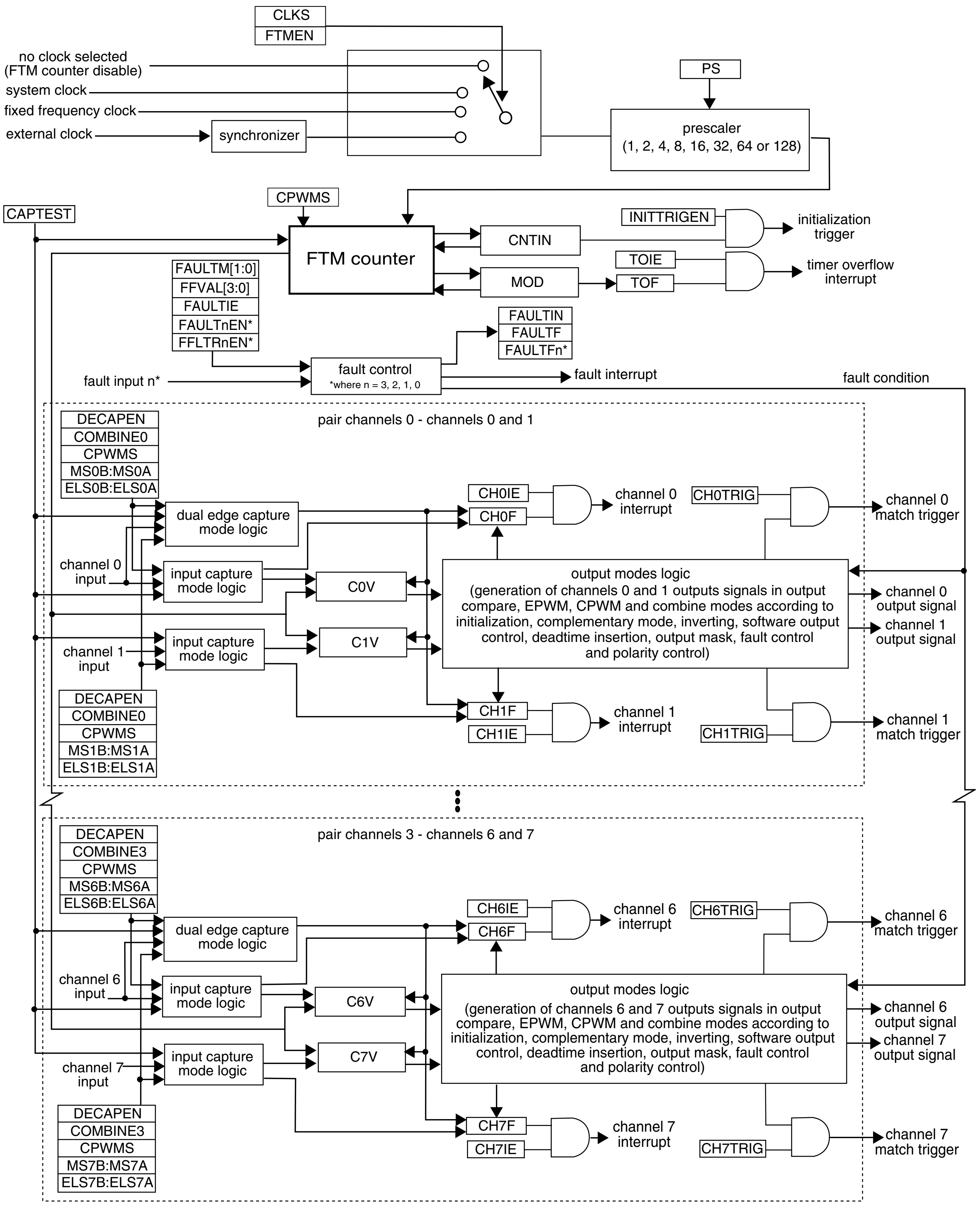


Figure 26-1. FTM block diagram

26.2 FTM signal descriptions

Table 26-1 shows the user-accessible signals for the FTM.

Table 26-1. FTM signal descriptions

Signal	Description	I/O	Function
EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I	The external clock input signal is used as the FTM counter clock if selected by CLKS[1:0] bits in the SC register. This clock signal must not exceed 1/4 of system clock frequency. The FTM counter prescaler selection and settings are also used when an external clock is selected.
CHn	FTM channel (n), where n can be 7-0	I/O	Each FTM channel can be configured to operate either as input or output. The direction associated with each channel, input or output, is selected according to the mode assigned for that channel.
FAULTj	Fault input (j), where j can be 3-0	I	The fault input signals are used to control the CHn channel output state. If a fault is detected, the FAULTj signal is asserted and the channel output is put in a safe state. The behavior of the fault logic is defined by the FAULTM[1:0] control bits in the MODE register and FAULTEN bit in the COMBINEm register. Note that each FAULTj input may affect all channels selectively since FAULTM[1:0] and FAULTEN control bits are defined for each pair of channels. Because there are several FAULTj inputs, maximum of 4 for the FTM module, each one of these inputs is activated by the FAULTjEN bit in the FLTCTRL register.

26.3 Memory map and register definition

26.3.1 Memory map

This section presents a high-level summary of the FTM registers and how they are mapped.

The registers and bits of an unavailable function in the FTM remain in the memory map and in the reset value, but they have no active function.

Note

Do not write in the region from the CNTIN register through the PWMLOAD register when FTMEN = 0.

26.3.2 Register descriptions

Accesses to reserved addresses result in transfer errors. Registers for absent channels are considered reserved.

FTM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_8000	Status And Control (FTM0_SC)	32	R/W	0000_0000h	26.3.3/351
4003_8004	Counter (FTM0_CNT)	32	R/W	0000_0000h	26.3.4/353
4003_8008	Modulo (FTM0_MOD)	32	R/W	0000_0000h	26.3.5/353
4003_800C	Channel (n) Status And Control (FTM0_C0SC)	32	R/W	0000_0000h	26.3.6/354
4003_8010	Channel (n) Value (FTM0_C0V)	32	R/W	0000_0000h	26.3.7/356
4003_8014	Channel (n) Status And Control (FTM0_C1SC)	32	R/W	0000_0000h	26.3.6/354
4003_8018	Channel (n) Value (FTM0_C1V)	32	R/W	0000_0000h	26.3.7/356
4003_801C	Channel (n) Status And Control (FTM0_C2SC)	32	R/W	0000_0000h	26.3.6/354
4003_8020	Channel (n) Value (FTM0_C2V)	32	R/W	0000_0000h	26.3.7/356
4003_8024	Channel (n) Status And Control (FTM0_C3SC)	32	R/W	0000_0000h	26.3.6/354
4003_8028	Channel (n) Value (FTM0_C3V)	32	R/W	0000_0000h	26.3.7/356
4003_802C	Channel (n) Status And Control (FTM0_C4SC)	32	R/W	0000_0000h	26.3.6/354
4003_8030	Channel (n) Value (FTM0_C4V)	32	R/W	0000_0000h	26.3.7/356
4003_8034	Channel (n) Status And Control (FTM0_C5SC)	32	R/W	0000_0000h	26.3.6/354
4003_8038	Channel (n) Value (FTM0_C5V)	32	R/W	0000_0000h	26.3.7/356
4003_803C	Channel (n) Status And Control (FTM0_C6SC)	32	R/W	0000_0000h	26.3.6/354
4003_8040	Channel (n) Value (FTM0_C6V)	32	R/W	0000_0000h	26.3.7/356
4003_8044	Channel (n) Status And Control (FTM0_C7SC)	32	R/W	0000_0000h	26.3.6/354
4003_8048	Channel (n) Value (FTM0_C7V)	32	R/W	0000_0000h	26.3.7/356
4003_804C	Counter Initial Value (FTM0_CNTIN)	32	R/W	0000_0000h	26.3.8/357
4003_8050	Capture And Compare Status (FTM0_STATUS)	32	R/W	0000_0000h	26.3.9/358
4003_8054	Features Mode Selection (FTM0_MODE)	32	R/W	0000_0004h	26.3.10/360
4003_8058	Synchronization (FTM0_SYNC)	32	R/W	0000_0000h	26.3.11/361
4003_805C	Initial State For Channels Output (FTM0_OUTINIT)	32	R/W	0000_0000h	26.3.12/364
4003_8060	Output Mask (FTM0_OUTMASK)	32	R/W	0000_0000h	26.3.13/365
4003_8064	Function For Linked Channels (FTM0_COMBINE)	32	R/W	0000_0000h	26.3.14/367
4003_8068	Deadtime Insertion Control (FTM0_DEADTIME)	32	R/W	0000_0000h	26.3.15/372
4003_806C	FTM External Trigger (FTM0_EXTTRIG)	32	R/W	0000_0000h	26.3.16/373
4003_8070	Channels Polarity (FTM0_POL)	32	R/W	0000_0000h	26.3.17/375
4003_8074	Fault Mode Status (FTM0_FMS)	32	R/W	0000_0000h	26.3.18/377
4003_8078	Input Capture Filter Control (FTM0_FILTER)	32	R/W	0000_0000h	26.3.19/379
4003_807C	Fault Control (FTM0_FLTCTRL)	32	R/W	0000_0000h	26.3.20/380
4003_8084	Configuration (FTM0_CONF)	32	R/W	0000_0000h	26.3.21/382
4003_8088	FTM Fault Input Polarity (FTM0_FLTPOL)	32	R/W	0000_0000h	26.3.22/383

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_808C	Synchronization Configuration (FTM0_SYNCONF)	32	R/W	0000_0000h	26.3.23/385
4003_8090	FTM Inverting Control (FTM0_INVCTRL)	32	R/W	0000_0000h	26.3.24/387
4003_8094	FTM Software Output Control (FTM0_SWOCTRL)	32	R/W	0000_0000h	26.3.25/388
4003_8098	FTM PWM Load (FTM0_PWMLOAD)	32	R/W	0000_0000h	26.3.26/390
4003_A000	Status And Control (FTM2_SC)	32	R/W	0000_0000h	26.3.3/351
4003_A004	Counter (FTM2_CNT)	32	R/W	0000_0000h	26.3.4/353
4003_A008	Modulo (FTM2_MOD)	32	R/W	0000_0000h	26.3.5/353
4003_A00C	Channel (n) Status And Control (FTM2_C0SC)	32	R/W	0000_0000h	26.3.6/354
4003_A010	Channel (n) Value (FTM2_C0V)	32	R/W	0000_0000h	26.3.7/356
4003_A014	Channel (n) Status And Control (FTM2_C1SC)	32	R/W	0000_0000h	26.3.6/354
4003_A018	Channel (n) Value (FTM2_C1V)	32	R/W	0000_0000h	26.3.7/356
4003_A01C	Channel (n) Status And Control (FTM2_C2SC)	32	R/W	0000_0000h	26.3.6/354
4003_A020	Channel (n) Value (FTM2_C2V)	32	R/W	0000_0000h	26.3.7/356
4003_A024	Channel (n) Status And Control (FTM2_C3SC)	32	R/W	0000_0000h	26.3.6/354
4003_A028	Channel (n) Value (FTM2_C3V)	32	R/W	0000_0000h	26.3.7/356
4003_A02C	Channel (n) Status And Control (FTM2_C4SC)	32	R/W	0000_0000h	26.3.6/354
4003_A030	Channel (n) Value (FTM2_C4V)	32	R/W	0000_0000h	26.3.7/356
4003_A034	Channel (n) Status And Control (FTM2_C5SC)	32	R/W	0000_0000h	26.3.6/354
4003_A038	Channel (n) Value (FTM2_C5V)	32	R/W	0000_0000h	26.3.7/356
4003_A03C	Channel (n) Status And Control (FTM2_C6SC)	32	R/W	0000_0000h	26.3.6/354
4003_A040	Channel (n) Value (FTM2_C6V)	32	R/W	0000_0000h	26.3.7/356
4003_A044	Channel (n) Status And Control (FTM2_C7SC)	32	R/W	0000_0000h	26.3.6/354
4003_A048	Channel (n) Value (FTM2_C7V)	32	R/W	0000_0000h	26.3.7/356
4003_A04C	Counter Initial Value (FTM2_CNTIN)	32	R/W	0000_0000h	26.3.8/357
4003_A050	Capture And Compare Status (FTM2_STATUS)	32	R/W	0000_0000h	26.3.9/358
4003_A054	Features Mode Selection (FTM2_MODE)	32	R/W	0000_0004h	26.3.10/360
4003_A058	Synchronization (FTM2_SYNC)	32	R/W	0000_0000h	26.3.11/361
4003_A05C	Initial State For Channels Output (FTM2_OUTINIT)	32	R/W	0000_0000h	26.3.12/364
4003_A060	Output Mask (FTM2_OUTMASK)	32	R/W	0000_0000h	26.3.13/365
4003_A064	Function For Linked Channels (FTM2_COMBINE)	32	R/W	0000_0000h	26.3.14/367
4003_A068	Deadtime Insertion Control (FTM2_DEADTIME)	32	R/W	0000_0000h	26.3.15/372
4003_A06C	FTM External Trigger (FTM2_EXTTRIG)	32	R/W	0000_0000h	26.3.16/373
4003_A070	Channels Polarity (FTM2_POL)	32	R/W	0000_0000h	26.3.17/375
4003_A074	Fault Mode Status (FTM2_FMS)	32	R/W	0000_0000h	26.3.18/377
4003_A078	Input Capture Filter Control (FTM2_FILTER)	32	R/W	0000_0000h	26.3.19/379
4003_A07C	Fault Control (FTM2_FLTCTRL)	32	R/W	0000_0000h	26.3.20/380
4003_A084	Configuration (FTM2_CONF)	32	R/W	0000_0000h	26.3.21/382
4003_A088	FTM Fault Input Polarity (FTM2_FLTPOL)	32	R/W	0000_0000h	26.3.22/383

Table continues on the next page...

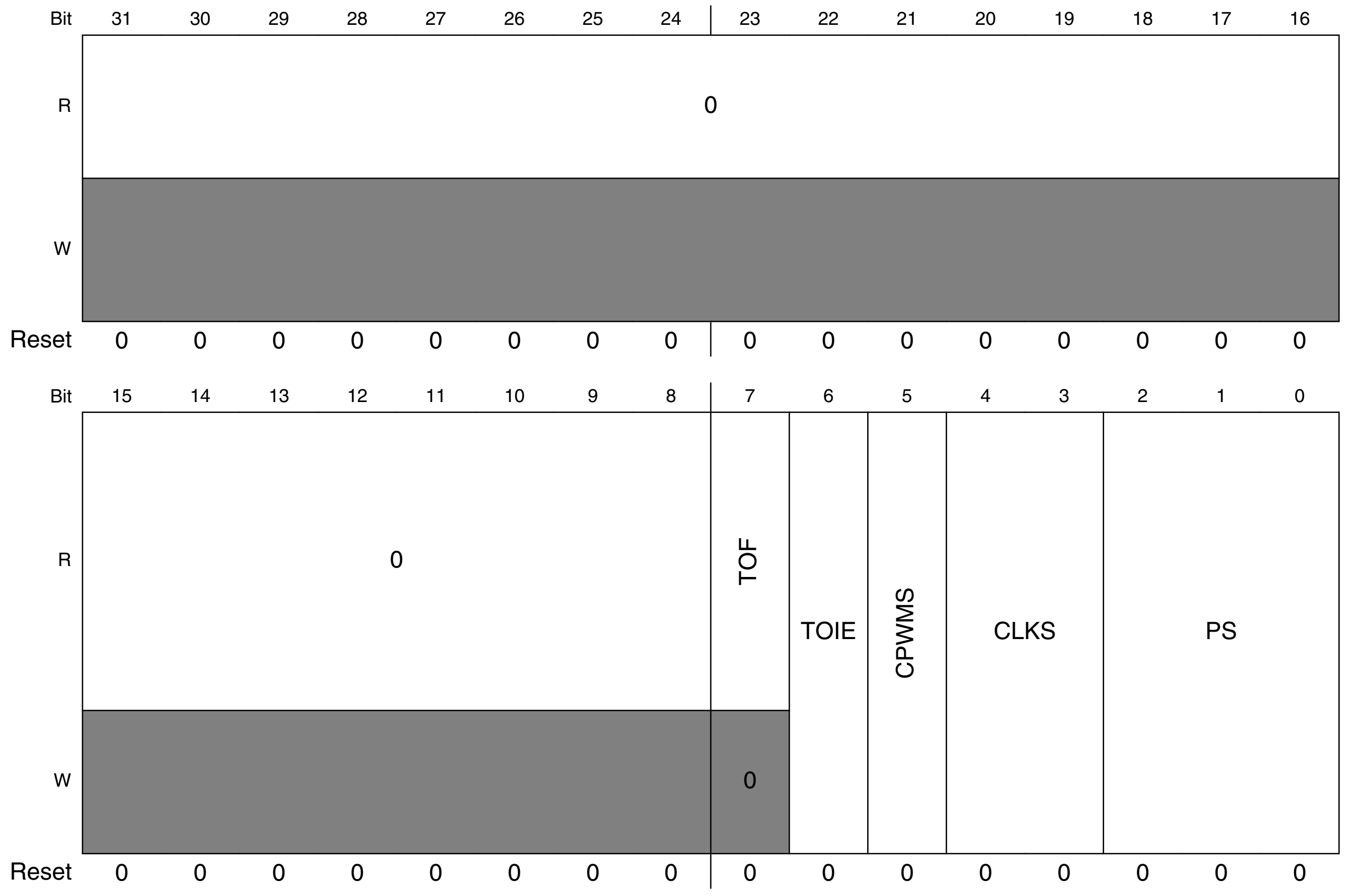
FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_A08C	Synchronization Configuration (FTM2_SYNCONF)	32	R/W	0000_0000h	26.3.23/385
4003_A090	FTM Inverting Control (FTM2_INVCTRL)	32	R/W	0000_0000h	26.3.24/387
4003_A094	FTM Software Output Control (FTM2_SWOCTRL)	32	R/W	0000_0000h	26.3.25/388
4003_A098	FTM PWM Load (FTM2_PWMLOAD)	32	R/W	0000_0000h	26.3.26/390

26.3.3 Status And Control (FTMx_SC)

SC contains the overflow status flag and control bits used to configure the interrupt enable, FTM configuration, clock source, and prescaler factor. These controls relate to all channels within this module.

Address: Base address + 0h offset



FTMx_SC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TOF	Timer Overflow Flag Set by hardware when the FTM counter passes the value in the MOD register. The TOF bit is cleared by reading the SC register while TOF is set and then writing a 0 to TOF bit. Writing a 1 to TOF has no effect. If another FTM overflow occurs between the read and write operations, the write operation has no effect; therefore, TOF remains set indicating an overflow has occurred. In this case, a TOF interrupt request is not lost due to the clearing sequence for a previous TOF. 0 FTM counter has not overflowed. 1 FTM counter has overflowed.
6 TOIE	Timer Overflow Interrupt Enable Enables FTM overflow interrupts. 0 Disable TOF interrupts. Use software polling. 1 Enable TOF interrupts. An interrupt is generated when TOF equals one.
5 CPWMS	Center-Aligned PWM Select Selects CPWM mode. This mode configures the FTM to operate in Up-Down Counting mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 FTM counter operates in Up Counting mode. 1 FTM counter operates in Up-Down Counting mode.
4–3 CLKS	Clock Source Selection Selects one of the three FTM counter clock sources. This field is write protected. It can be written only when MODE[WPDIS] = 1. 00 No clock selected. This in effect disables the FTM counter. 01 System clock 10 Fixed frequency clock 11 External clock
PS	Prescale Factor Selection Selects one of 8 division factors for the clock source selected by CLKS. The new prescaler factor affects the clock source on the next system clock cycle after the new value is updated into the register bits. This field is write protected. It can be written only when MODE[WPDIS] = 1. 000 Divide by 1 001 Divide by 2 010 Divide by 4 011 Divide by 8 100 Divide by 16 101 Divide by 32 110 Divide by 64 111 Divide by 128

26.3.4 Counter (FTMx_CNT)

The CNT register contains the FTM counter value.

Reset clears the CNT register. Writing any value to COUNT updates the counter with its initial value, CNTIN.

When Debug is active, the FTM counter is frozen. This is the value that you may read.

Address: Base address + 4h offset

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

FTMx_CNT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
COUNT	Counter Value

26.3.5 Modulo (FTMx_MOD)

The Modulo register contains the modulo value for the FTM counter. After the FTM counter reaches the modulo value, the overflow flag (TOF) becomes set at the next clock, and the next value of FTM counter depends on the selected counting method; see Counter.

Writing to the MOD register latches the value into a buffer. The MOD register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the SC register whether Debug is active or not.

Initialize the FTM counter, by writing to CNT, before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

Address: Base address + 8h offset

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

FTMx_MOD field descriptions

Field	Description
31–16 Reserved	This field is reserved.
MOD	Modulo Value

26.3.6 Channel (n) Status And Control (FTMx_C*n*SC)

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

Table 26-69. Mode, edge, and level selection

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	X	X	XX	0	Pin not used for FTM—revert the channel pin to general purpose I/O or other peripheral control	
0	0	0	0	1	Input Capture	Capture on Rising Edge Only
				10		Capture on Falling Edge Only
				11		Capture on Rising or Falling Edge
			1	1	Output Compare	Toggle Output on match
				10		Clear Output on match
				11		Set Output on match
			1X	10	Edge-Aligned PWM	High-true pulses (clear Output on match)
				X1		Low-true pulses (set Output on match)
		1	XX	10	Center-Aligned PWM	High-true pulses (clear Output on match-up)
				X1		Low-true pulses (set Output on match-up)

Table continues on the next page...

Table 26-69. Mode, edge, and level selection (continued)

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
	1	0	XX	10	Combine PWM	High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
				X1		Low-true pulses (clear on channel (n) match, and set on channel (n +1) match)
1	0	0	X0	See the following table (Table 26-8).	Dual Edge Capture	One-Shot Capture mode
			X1			Continuous Capture mode

Table 26-70. Dual Edge Capture mode — edge polarity selection

ELSnB	ELSnA	Channel Port Enable	Detected Edges
0	0	Disabled	No edge
0	1	Enabled	Rising edge
1	0	Enabled	Falling edge
1	1	Enabled	Rising and falling edges

Address: Base address + Ch offset + (8d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								CHF	CHIE	MSB	MSA	ELSB	ELSA	0	0
W	▨	▨	▨	▨	▨	▨	▨	▨	0						▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_CnSC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CHF	Channel Flag Set by hardware when an event occurs on the channel. CHF is cleared by reading the CSC register while CHnF is set and then writing a 0 to the CHF bit. Writing a 1 to CHF has no effect.

Table continues on the next page...

FTMx_*Cn*SC field descriptions (continued)

Field	Description
	If another event occurs between the read and write operations, the write operation has no effect; therefore, CHF remains set indicating an event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF. 0 No channel event has occurred. 1 A channel event has occurred.
6 CHIE	Channel Interrupt Enable Enables channel interrupts. 0 Disable channel interrupts. Use software polling. 1 Enable channel interrupts.
5 MSB	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 26-7. This field is write protected. It can be written only when MODE[WPDIS] = 1.
4 MSA	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 26-7. This field is write protected. It can be written only when MODE[WPDIS] = 1.
3 ELSB	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. See Table 26-7. This field is write protected. It can be written only when MODE[WPDIS] = 1.
2 ELSA	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. See Table 26-7. This field is write protected. It can be written only when MODE[WPDIS] = 1.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

26.3.7 Channel (n) Value (FTMx_*Cn*V)

These registers contain the captured FTM counter value for the input modes or the match value for the output modes.

In Input Capture, Capture Test, and Dual Edge Capture modes, any write to a CnV register is ignored.

In output modes, writing to a CnV register latches the value into a buffer. A CnV register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the CnSC register whether Debug mode is active or not.

Address: Base address + 10h offset + (8d × i), where i=0d to 7d

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

FTMx_*Cn*V field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
VAL	Channel Value Captured FTM counter value of the input modes or the match value for the output modes

26.3.8 Counter Initial Value (FTMx_CNTIN)

The Counter Initial Value register contains the initial value for the FTM counter.

Writing to the CNTIN register latches the value into a buffer. The CNTIN register is updated with the value of its write buffer according to Registers updated from write buffers.

When the FTM clock is initially selected, by writing a non-zero value to the CLKS bits, the FTM counter starts with the value 0x0000. To avoid this behavior, before the first write to select the FTM clock, write the new value to the the CNTIN register and then initialize the FTM counter by writing any value to the CNT register.

Address: Base address + 4Ch offset

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

FTMx_CNTIN field descriptions

Field	Description
31–16 Reserved	This field is reserved.
INIT	Initial Value Of The FTM Counter

26.3.9 Capture And Compare Status (FTMx_STATUS)

The STATUS register contains a copy of the status flag CHnF bit in CnSC for each FTM channel for software convenience.

Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by reading STATUS followed by writing 0x00 to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. CHnF is cleared by reading STATUS while CHnF is set and then writing a 0 to the CHnF bit. Writing a 1 to CHnF has no effect.

If another event occurs between the read and write operations, the write operation has no effect; therefore, CHnF remains set indicating an event has occurred. In this case, a CHnF interrupt request is not lost due to the clearing sequence for a previous CHnF.

NOTE

The STATUS register should be used only in Combine mode.

Address: Base address + 50h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W									0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_STATUS field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CH7F	Channel 7 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
6 CH6F	Channel 6 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
5 CH5F	Channel 5 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
4 CH4F	Channel 4 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
3 CH3F	Channel 3 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
2 CH2F	Channel 2 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
1 CH1F	Channel 1 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
0 CH0F	Channel 0 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.

26.3.10 Features Mode Selection (FTMx_MODE)

This register contains the global enable bit for FTM-specific features and the control bits used to configure:

- Fault control mode and interrupt
- Capture Test mode
- PWM synchronization
- Write protection
- Channel output initialization

These controls relate to all channels within this module.

Address: Base address + 54h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								FAULTIE	FAULTM		CAPTEST	PWMSYNC	WPDIS	INIT	FTMEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

FTMx_MODE field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 FAULTIE	Fault Interrupt Enable Enables the generation of an interrupt when a fault is detected by FTM and the FTM fault control is enabled. 0 Fault control interrupt is disabled. 1 Fault control interrupt is enabled.
6–5 FAULTM	Fault Control Mode Defines the FTM fault control mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 00 Fault control is disabled for all channels. 01 Fault control is enabled for even channels only (channels 0, 2, 4, and 6), and the selected mode is the manual fault clearing.

Table continues on the next page...

FTMx_MODE field descriptions (continued)

Field	Description
	10 Fault control is enabled for all channels, and the selected mode is the manual fault clearing. 11 Fault control is enabled for all channels, and the selected mode is the automatic fault clearing.
4 CAPTEST	Capture Test Mode Enable Enables the capture test mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Capture test mode is disabled. 1 Capture test mode is enabled.
3 PWMSYNC	PWM Synchronization Mode Selects which triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization. See PWM synchronization. The PWMSYNC bit configures the synchronization when SYNCMODE is 0. 0 No restrictions. Software and hardware triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization. 1 Software trigger can only be used by MOD and CnV synchronization, and hardware triggers can only be used by OUTMASK and FTM counter synchronization.
2 WPDIS	Write Protection Disable When write protection is enabled (WPDIS = 0), write protected bits cannot be written. When write protection is disabled (WPDIS = 1), write protected bits can be written. The WPDIS bit is the negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect. 0 Write protection is enabled. 1 Write protection is disabled.
1 INIT	Initialize The Channels Output When a 1 is written to INIT bit the channels output is initialized according to the state of their corresponding bit in the OUTINIT register. Writing a 0 to INIT bit has no effect. The INIT bit is always read as 0.
0 FTMEN	FTM Enable This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Only the TPM-compatible registers (first set of registers) can be used without any restriction. Do not use the FTM-specific registers. 1 All registers including the FTM-specific registers (second set of registers) are available for use with no restrictions.

26.3.11 Synchronization (FTMx_SYNC)

This register configures the PWM synchronization.

A synchronization event can perform the synchronized update of MOD, CV, and OUTMASK registers with the value of their write buffer and the FTM counter initialization.

NOTE

The software trigger, SWSYNC bit, and hardware triggers TRIG0, TRIG1, and TRIG2 bits have a potential conflict if used together when SYNCMODE = 0. Use only hardware or software triggers but not both at the same time, otherwise unpredictable behavior is likely to happen.

The selection of the loading point, CNTMAX and CNTMIN bits, is intended to provide the update of MOD, CNTIN, and CnV registers across all enabled channels simultaneously. The use of the loading point selection together with SYNCMODE = 0 and hardware trigger selection, TRIG0, TRIG1, or TRIG2 bits, is likely to result in unpredictable behavior.

The synchronization event selection also depends on the PWMSYNC (MODE register) and SYNCMODE (SYNCONF register) bits. See PWM synchronization.

Address: Base address + 58h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SWSYNC	TRIG2	TRIG1	TRIG0	SYNCHOM	REINIT	CNTMAX	CNTMIN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SYNC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SWSYNC	PWM Synchronization Software Trigger Selects the software trigger as the PWM synchronization trigger. The software trigger happens when a 1 is written to SWSYNC bit. 0 Software trigger is not selected. 1 Software trigger is selected.
6 TRIG2	PWM Synchronization Hardware Trigger 2

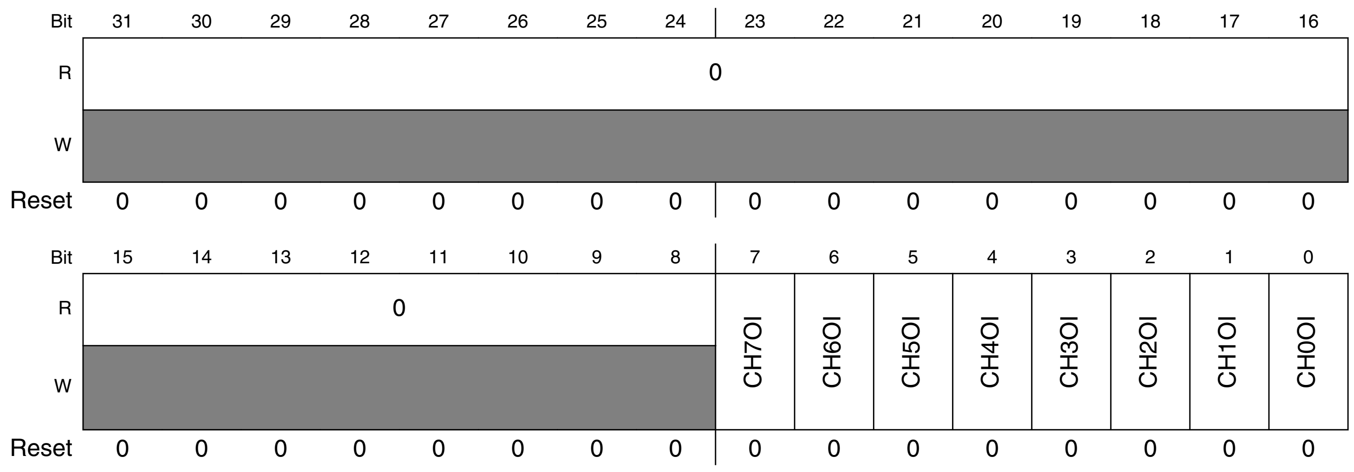
Table continues on the next page...

FTMx_SYNC field descriptions (continued)

Field	Description
	Enables hardware trigger 2 to the PWM synchronization. Hardware trigger 2 happens when a rising edge is detected at the trigger 2 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
5 TRIG1	PWM Synchronization Hardware Trigger 1 Enables hardware trigger 1 to the PWM synchronization. Hardware trigger 1 happens when a rising edge is detected at the trigger 1 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
4 TRIG0	PWM Synchronization Hardware Trigger 0 Enables hardware trigger 0 to the PWM synchronization. Hardware trigger 0 occurs when a rising edge is detected at the trigger 0 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
3 SYNCHOM	Output Mask Synchronization Selects when the OUTMASK register is updated with the value of its buffer. 0 OUTMASK register is updated with the value of its buffer in all rising edges of the system clock. 1 OUTMASK register is updated with the value of its buffer only by the PWM synchronization.
2 REINIT	FTM Counter Reinitialization By Synchronization (FTM counter synchronization) Determines if the FTM counter is reinitialized when the selected trigger for the synchronization is detected. The REINIT bit configures the synchronization when SYNCMODE is zero. 0 FTM counter continues to count normally. 1 FTM counter is updated with its initial value when the selected trigger is detected.
1 CNTMAX	Maximum Loading Point Enable Selects the maximum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMAX is 1, the selected loading point is when the FTM counter reaches its maximum value (MOD register). 0 The maximum loading point is disabled. 1 The maximum loading point is enabled.
0 CNTMIN	Minimum Loading Point Enable Selects the minimum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMIN is one, the selected loading point is when the FTM counter reaches its minimum value (CNTIN register). 0 The minimum loading point is disabled. 1 The minimum loading point is enabled.

26.3.12 Initial State For Channels Output (FTMx_OUTINIT)

Address: Base address + 5Ch offset



FTMx_OUTINIT field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CH7OI	Channel 7 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
6 CH6OI	Channel 6 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
5 CH5OI	Channel 5 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
4 CH4OI	Channel 4 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
3 CH3OI	Channel 3 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs.

Table continues on the next page...

FTMx_OUTINIT field descriptions (continued)

Field	Description
	0 The initialization value is 0. 1 The initialization value is 1.
2 CH2OI	Channel 2 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
1 CH1OI	Channel 1 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
0 CH0OI	Channel 0 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.

26.3.13 Output Mask (FTMx_OUTMASK)

This register provides a mask for each FTM channel. The mask of a channel determines if its output responds, that is, it is masked or not, when a match occurs. This feature is used for BLDC control where the PWM signal is presented to an electric motor at specific times to provide electronic commutation.

Any write to the OUTMASK register, stores the value in its write buffer. The register is updated with the value of its write buffer according to PWM synchronization.

Address: Base address + 60h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								CH7OM	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM	CH0OM
W	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_OUTMASK field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CH7OM	Channel 7 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
6 CH6OM	Channel 6 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
5 CH5OM	Channel 5 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
4 CH4OM	Channel 4 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
3 CH3OM	Channel 3 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
2 CH2OM	Channel 2 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
1 CH1OM	Channel 1 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
0 CH0OM	Channel 0 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.

26.3.14 Function For Linked Channels (FTMx_COMBINE)

This register contains the control bits used to configure the fault control, synchronization, deadtime insertion, Dual Edge Capture mode, Complementary, and Combine mode for each pair of channels (n) and (n+1), where n equals 0, 2, 4, and 6.

Address: Base address + 64h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	FAULTEN3	SYNCEN3	DTEN3	DECAP3	DECAPEN3	COMP3	COMBINE3	0	FAULTEN2	SYNCEN2	DTEN2	DECAP2	DECAPEN2	COMP2	COMBINE2
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FAULTEN1	SYNCEN1	DTEN1	DECAP1	DECAPEN1	COMP1	COMBINE1	0	FAULTEN0	SYNCEN0	DTEN0	DECAP0	DECAPEN0	COMP0	COMBINE0
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_COMBINE field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FAULTEN3	Fault Control Enable For n = 6 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
29 SYNCEN3	Synchronization Enable For n = 6 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
28 DTEN3	Deadtime Enable For n = 6 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
27 DECAP3	Dual Edge Capture Mode Captures For n = 6 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when FTMEN = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
26 DECAPEN3	Dual Edge Capture Mode Enable For n = 6 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 26-7. This field applies only when FTMEN = 1. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
25 COMP3	Complement Of Channel (n) for n = 6 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
24 COMBINE3	Combine Channels For n = 6 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 FAULTEN2	Fault Control Enable For n = 4 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
21 SYNCEN2	Synchronization Enable For n = 4 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
20 DTEN2	Deadtime Enable For n = 4 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
19 DECAP2	Dual Edge Capture Mode Captures For n = 4 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when FTMEN = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
18 DECAPEN2	Dual Edge Capture Mode Enable For n = 4 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 26-7. This field applies only when FTMEN = 1. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
17 COMP2	Complement Of Channel (n) For n = 4 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
16 COMBINE2	Combine Channels For n = 4 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 FAULTEN1	Fault Control Enable For n = 2 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
13 SYNCEN1	Synchronization Enable For n = 2 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
12 DTEN1	Deadtime Enable For n = 2 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
11 DECAP1	Dual Edge Capture Mode Captures For n = 2 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when FTMEN = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if Dual Edge Capture – One-Shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
10 DECAPEN1	Dual Edge Capture Mode Enable For n = 2 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 26-7. This field applies only when FTMEN = 1. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
9 COMP1	Complement Of Channel (n) For n = 2 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
8 COMBINE1	Combine Channels For n = 2 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FAULTEN0	Fault Control Enable For n = 0 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
5 SYNCEN0	Synchronization Enable For n = 0 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
4 DTEN0	Deadtime Enable For n = 0 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
3 DECAP0	Dual Edge Capture Mode Captures For n = 0 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when FTMEN = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
2 DECAPEN0	Dual Edge Capture Mode Enable For n = 0 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 26-7. This field applies only when FTMEN = 1. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
1 COMP0	Complement Of Channel (n) For n = 0 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
0 COMBINE0	Combine Channels For n = 0 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.

26.3.15 Deadtime Insertion Control (FTMx_DEADTIME)

This register selects the deadtime prescaler factor and deadtime value. All FTM channels use this clock prescaler and this deadtime value for the deadtime insertion.

Address: Base address + 68h offset

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

FTMx_DEADTIME field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 DTPS	Deadtime Prescaler Value Selects the division factor of the system clock. This prescaled clock is used by the deadtime counter. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0x Divide the system clock by 1. 10 Divide the system clock by 4. 11 Divide the system clock by 16.
DTVAL	Deadtime Value Selects the deadtime insertion value for the deadtime counter. The deadtime counter is clocked by a scaled version of the system clock. See the description of DTPS. Deadtime insert value = (DTPS × DTVAL). DTVAL selects the number of deadtime counts inserted as follows: When DTVAL is 0, no counts are inserted. When DTVAL is 1, 1 count is inserted. When DTVAL is 2, 2 counts are inserted. This pattern continues up to a possible 63 counts. This field is write protected. It can be written only when MODE[WPDIS] = 1.

26.3.16 FTM External Trigger (FTMx_EXTTRIG)

This register:

- Indicates when a channel trigger was generated
- Enables the generation of a trigger when the FTM counter is equal to its initial value
- Selects which channels are used in the generation of the channel triggers

Several channels can be selected to generate multiple triggers in one PWM period.

Channels 6 and 7 are not used to generate channel triggers.

Address: Base address + 6Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								TRIGF	INITTRIGEN	CH1TRIG	CH0TRIG	CH5TRIG	CH4TRIG	CH3TRIG	CH2TRIG
W									0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_EXTTRIG field descriptions

Field	Description
31–8 Reserved	This field is reserved.
7 TRIGF	Channel Trigger Flag Set by hardware when a channel trigger is generated. Clear TRIGF by reading EXTTRIG while TRIGF is set and then writing a 0 to TRIGF. Writing a 1 to TRIGF has no effect. If another channel trigger is generated before the clearing sequence is completed, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF. 0 No channel trigger was generated. 1 A channel trigger was generated.
6 INITTRIGEN	Initialization Trigger Enable Enables the generation of the trigger when the FTM counter is equal to the CNTIN register. 0 The generation of initialization trigger is disabled. 1 The generation of initialization trigger is enabled.
5 CH1TRIG	Channel 1 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
4 CH0TRIG	Channel 0 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
3 CH5TRIG	Channel 5 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
2 CH4TRIG	Channel 4 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
1 CH3TRIG	Channel 3 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
0 CH2TRIG	Channel 2 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register.

Table continues on the next page...

FTMx_EXTTRIG field descriptions (continued)

Field	Description
	0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.

26.3.17 Channels Polarity (FTMx_POL)

This register defines the output polarity of the FTM channels.

NOTE

The safe value that is driven in a channel output when the fault control is enabled and a fault condition is detected is the inactive state of the channel. That is, the safe value of a channel is the value of its POL bit.

Address: Base address + 70h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								POL7	POL6	POL5	POL4	POL3	POL2	POL1	POL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_POL field descriptions

Field	Description
31–8 Reserved	This field is reserved.
7 POL7	Channel 7 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
6 POL6	Channel 6 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
5 POL5	Channel 5 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

FTMx_POL field descriptions (continued)

Field	Description
	0 The channel polarity is active high. 1 The channel polarity is active low.
4 POL4	Channel 4 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
3 POL3	Channel 3 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
2 POL2	Channel 2 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
1 POL1	Channel 1 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
0 POL0	Channel 0 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.

26.3.18 Fault Mode Status (FTMx_FMS)

This register contains the fault detection flags, write protection enable bit, and the logic OR of the enabled fault inputs.

Address: Base address + 74h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								FAULTF	WPEN	FAULTIN	0	FAULTF3	FAULTF2	FAULTF1	FAULTF0
W									0				0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_FMS field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 FAULTF	Fault Detection Flag Represents the logic OR of the individual FAULTFj bits where j = 3, 2, 1, 0. Clear FAULTF by reading the FMS register while FAULTF is set and then writing a 0 to FAULTF while there is no existing fault condition at the enabled fault inputs. Writing a 1 to FAULTF has no effect. If another fault condition is detected in an enabled fault input before the clearing sequence is completed, the sequence is reset so FAULTF remains set after the clearing sequence is completed for the earlier fault condition. FAULTF is also cleared when FAULTFj bits are cleared individually. 0 No fault condition was detected. 1 A fault condition was detected.

Table continues on the next page...

FTMx_FMS field descriptions (continued)

Field	Description
6 WPEN	Write Protection Enable The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect. 0 Write protection is disabled. Write protected bits can be written. 1 Write protection is enabled. Write protected bits cannot be written.
5 FAULTIN	Fault Inputs Represents the logic OR of the enabled fault inputs after their filter (if their filter is enabled) when fault control is enabled. 0 The logic OR of the enabled fault inputs is 0. 1 The logic OR of the enabled fault inputs is 1.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FAULTF3	Fault Detection Flag 3 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF3 by reading the FMS register while FAULTF3 is set and then writing a 0 to FAULTF3 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF3 has no effect. FAULTF3 bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF3 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.
2 FAULTF2	Fault Detection Flag 2 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF2 by reading the FMS register while FAULTF2 is set and then writing a 0 to FAULTF2 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF2 has no effect. FAULTF2 bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF2 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.
1 FAULTF1	Fault Detection Flag 1 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF1 by reading the FMS register while FAULTF1 is set and then writing a 0 to FAULTF1 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF1 has no effect. FAULTF1 bit is also cleared when FAULTF bit is cleared.

Table continues on the next page...

FTMx_FMS field descriptions (continued)

Field	Description
	If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF1 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.
0 FAULTF0	Fault Detection Flag 0 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF0 by reading the FMS register while FAULTF0 is set and then writing a 0 to FAULTF0 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF0 has no effect. FAULTF0 bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF0 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.

26.3.19 Input Capture Filter Control (FTMx_FILTER)

This register selects the filter value for the inputs of channels.

Channels 4, 5, 6 and 7 do not have an input filter.

NOTE

Writing to the FILTER register has immediate effect and must be done only when the channels 0, 1, 2, and 3 are not in input modes. Failure to do this could result in a missing valid signal.

Address: Base address + 78h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																CH3FVAL				CH2FVAL				CH1FVAL				CH0FVAL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_FILTER field descriptions

Field	Description
31–16 Reserved	This field is reserved.
15–12 CH3FVAL	Channel 3 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.

Table continues on the next page...

FTMx_FILTER field descriptions (continued)

Field	Description
11–8 CH2FVAL	Channel 2 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
7–4 CH1FVAL	Channel 1 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
CH0FVAL	Channel 0 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.

26.3.20 Fault Control (FTMx_FLTCTRL)

This register selects the filter value for the fault inputs, enables the fault inputs and the fault inputs filter.

Address: Base address + 7Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				FFVAL				FFLTR3EN	FFLTR2EN	FFLTR1EN	FFLTR0EN	FAULT3EN	FAULT2EN	FAULT1EN	FAULT0EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_FLTCTRL field descriptions

Field	Description
31–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–8 FFVAL	Fault Input Filter Selects the filter value for the fault inputs. The fault filter is disabled when the value is zero.

Table continues on the next page...

FTMx_FLTCTRL field descriptions (continued)

Field	Description
	NOTE: Writing to this field has immediate effect and must be done only when the fault control or all fault inputs are disabled. Failure to do this could result in a missing fault detection.
7 FFLTR3EN	Fault Input 3 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
6 FFLTR2EN	Fault Input 2 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
5 FFLTR1EN	Fault Input 1 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
4 FFLTR0EN	Fault Input 0 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
3 FAULT3EN	Fault Input 3 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.
2 FAULT2EN	Fault Input 2 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.
1 FAULT1EN	Fault Input 1 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

FTMx_FLTCTRL field descriptions (continued)

Field	Description
	0 Fault input is disabled. 1 Fault input is enabled.
0 FAULT0EN	Fault Input 0 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.

26.3.21 Configuration (FTMx_CONF)

This register selects the number of times that the FTM counter overflow should occur before the TOF bit to be set, the FTM behavior in Debug modes, the use of an external global time base, and the global time base signal generation.

Address: Base address + 84h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0					GTBEOUT	GTBEEN	0	BDMMODE		0	NUMTOF				
W	▨							▨			▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_CONF field descriptions

Field	Description
31–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 GTBEOUT	Global Time Base Output Enables the global time base signal generation to other FTMs. 0 A global time base signal generation is disabled. 1 A global time base signal generation is enabled.

Table continues on the next page...

FTMx_CONF field descriptions (continued)

Field	Description
9 GTBEEN	Global Time Base Enable Configures the FTM to use an external global time base signal that is generated by another FTM. 0 Use of an external global time base is disabled. 1 Use of an external global time base is enabled.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 BDMMODE	Debug Mode Selects the FTM behavior in Debug mode. See Debug mode.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
NUMTOF	TOF Frequency Selects the ratio between the number of counter overflows to the number of times the TOF bit is set. NUMTOF = 0: The TOF bit is set for each counter overflow. NUMTOF = 1: The TOF bit is set for the first counter overflow but not for the next overflow. NUMTOF = 2: The TOF bit is set for the first counter overflow but not for the next 2 overflows. NUMTOF = 3: The TOF bit is set for the first counter overflow but not for the next 3 overflows. This pattern continues up to a maximum of 31.

26.3.22 FTM Fault Input Polarity (FTMx_FLTPOL)

This register defines the fault inputs polarity.

Address: Base address + 88h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												FLT3POL	FLT2POL	FLT1POL	FLT0POL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_FLTPOL field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FLT3POL	Fault Input 3 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.
2 FLT2POL	Fault Input 2 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.
1 FLT1POL	Fault Input 1 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.
0 FLT0POL	Fault Input 0 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.

26.3.23 Synchronization Configuration (FTMx_SYNCONF)

This register selects the PWM synchronization configuration, SWOCTRL, INVCTRL and CNTIN registers synchronization, if FTM clears the TRIGj bit, where j = 0, 1, 2, when the hardware trigger j is detected.

Address: Base address + 8Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0											HWSOC	HWINVC	HWOM	HWWRBUF	HWRSTCNT
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			SWSOC	SWINVC	SWOM	SWWRBUF	SWRSTCNT	SYNCMODE	0	SWOC	INVC	0	CNTINC	0	HWTRIGMODE
W	▨									▨			▨		▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SYNCONF field descriptions

Field	Description
31–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 HWSOC	Software output control synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the SWOCTRL register synchronization. 1 A hardware trigger activates the SWOCTRL register synchronization.
19 HWINVC	Inverting control synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the INVCTRL register synchronization. 1 A hardware trigger activates the INVCTRL register synchronization.
18 HWOM	Output mask synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the OUTMASK register synchronization. 1 A hardware trigger activates the OUTMASK register synchronization.
17 HWWRBUF	MOD, CNTIN, and CV registers synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate MOD, CNTIN, and CV registers synchronization. 1 A hardware trigger activates MOD, CNTIN, and CV registers synchronization.
16 HWRSTCNT	FTM counter synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the FTM counter synchronization. 1 A hardware trigger activates the FTM counter synchronization.

Table continues on the next page...

FTMx_SYNCONF field descriptions (continued)

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 SWSOC	Software output control synchronization is activated by the software trigger. 0 The software trigger does not activate the SWOCTRL register synchronization. 1 The software trigger activates the SWOCTRL register synchronization.
11 SWINVC	Inverting control synchronization is activated by the software trigger. 0 The software trigger does not activate the INVCTRL register synchronization. 1 The software trigger activates the INVCTRL register synchronization.
10 SWOM	Output mask synchronization is activated by the software trigger. 0 The software trigger does not activate the OUTMASK register synchronization. 1 The software trigger activates the OUTMASK register synchronization.
9 SWWRBUF	MOD, CNTIN, and CV registers synchronization is activated by the software trigger. 0 The software trigger does not activate MOD, CNTIN, and CV registers synchronization. 1 The software trigger activates MOD, CNTIN, and CV registers synchronization.
8 SWRSTCNT	FTM counter synchronization is activated by the software trigger. 0 The software trigger does not activate the FTM counter synchronization. 1 The software trigger activates the FTM counter synchronization.
7 SYNCMODE	Synchronization Mode Selects the PWM Synchronization mode. 0 Legacy PWM synchronization is selected. 1 Enhanced PWM synchronization is selected.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SWOC	SWOCTRL Register Synchronization 0 SWOCTRL register is updated with its buffer value at all rising edges of system clock. 1 SWOCTRL register is updated with its buffer value by the PWM synchronization.
4 INVC	INVCTRL Register Synchronization 0 INVCTRL register is updated with its buffer value at all rising edges of system clock. 1 INVCTRL register is updated with its buffer value by the PWM synchronization.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CNTINC	CNTIN Register Synchronization 0 CNTIN register is updated with its buffer value at all rising edges of system clock. 1 CNTIN register is updated with its buffer value by the PWM synchronization.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 HWTRIGMODE	Hardware Trigger Mode

Table continues on the next page...

FTMx_SYNCONF field descriptions (continued)

Field	Description
	0 FTM clears the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2. 1 FTM does not clear the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2.

26.3.24 FTM Inverting Control (FTMx_INVCTRL)

This register controls when the channel (n) output becomes the channel (n+1) output, and channel (n+1) output becomes the channel (n) output. Each INVmEN bit enables the inverting operation for the corresponding pair channels m.

This register has a write buffer. The INVmEN bit is updated by the INVCTRL register synchronization.

Address: Base address + 90h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												INV3EN	INV2EN	INV1EN	INV0EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_INVCTRL field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 INV3EN	Pair Channels 3 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
2 INV2EN	Pair Channels 2 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
1 INV1EN	Pair Channels 1 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.

Table continues on the next page...

FTMx_INVCTRL field descriptions (continued)

Field	Description
0 INV0EN	Pair Channels 0 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.

26.3.25 FTM Software Output Control (FTMx_SWOCTRL)

This register enables software control of channel (n) output and defines the value forced to the channel (n) output:

- The CHnOC bits enable the control of the corresponding channel (n) output by software.
- The CHnOCV bits select the value that is forced at the corresponding channel (n) output.

This register has a write buffer. The fields are updated by the SWOCTRL register synchronization.

Address: Base address + 94h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	CH7OCV	CH6OCV	CH5OCV	CH4OCV	CH3OCV	CH2OCV	CH1OCV	CH0OCV	CH7OC	CH6OC	CH5OC	CH4OC	CH3OC	CH2OC	CH1OC	CH0OC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SWOCTRL field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CH7OCV	Channel 7 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
14 CH6OCV	Channel 6 Software Output Control Value

Table continues on the next page...

FTMx_SWOCTRL field descriptions (continued)

Field	Description
	0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
13 CH5OCV	Channel 5 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
12 CH4OCV	Channel 4 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
11 CH3OCV	Channel 3 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
10 CH2OCV	Channel 2 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
9 CH1OCV	Channel 1 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
8 CH0OCV	Channel 0 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
7 CH7OC	Channel 7 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
6 CH6OC	Channel 6 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
5 CH5OC	Channel 5 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
4 CH4OC	Channel 4 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
3 CH3OC	Channel 3 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
2 CH2OC	Channel 2 Software Output Control Enable

Table continues on the next page...

FTMx_SWOCTRL field descriptions (continued)

Field	Description
	0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
1 CH1OC	Channel 1 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
0 CH0OC	Channel 0 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.

26.3.26 FTM PWM Load (FTMx_PWMLOAD)

Enables the loading of the MOD, CNTIN, C(n)V, and C(n+1)V registers with the values of their write buffers when the FTM counter changes from the MOD register value to its next value or when a channel (j) match occurs. A match occurs for the channel (j) when FTM counter = C(j)V.

Address: Base address + 98h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						LDOK	0	CH7SEL	CH6SEL	CH5SEL	CH4SEL	CH3SEL	CH2SEL	CH1SEL	CH0SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_PWMLOAD field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 LDOK	Load Enable Enables the loading of the MOD, CNTIN, and CV registers with the values of their write buffers. 0 Loading updated values is disabled. 1 Loading updated values is enabled.

Table continues on the next page...

FTMx_PWMLOAD field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CH7SEL	Channel 7 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
6 CH6SEL	Channel 6 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
5 CH5SEL	Channel 5 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
4 CH4SEL	Channel 4 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
3 CH3SEL	Channel 3 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
2 CH2SEL	Channel 2 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
1 CH1SEL	Channel 1 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
0 CH0SEL	Channel 0 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.

26.4 Functional description

The notation used in this document to represent the counters and the generation of the signals is shown in the following figure.

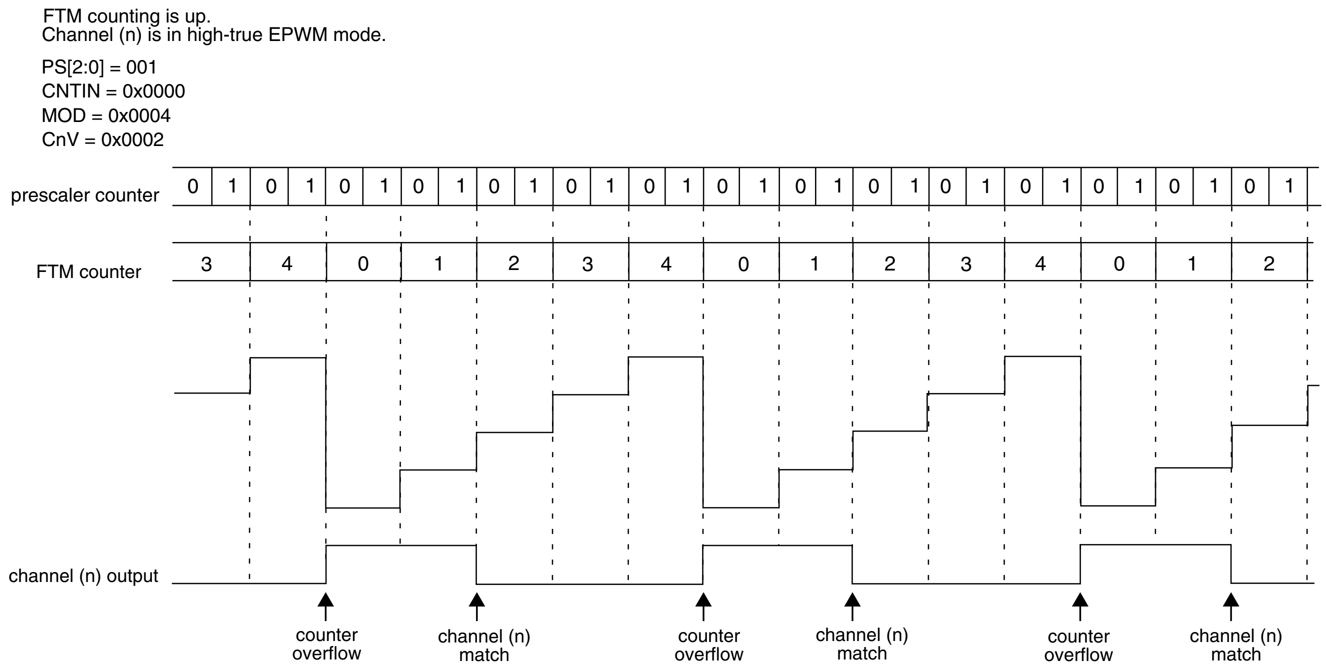


Figure 26-122. Notation used

26.4.1 Clock source

The FTM has only one clock domain: the system clock.

26.4.1.1 Counter clock source

The CLKS[1:0] bits in the SC register select one of three possible clock sources for the FTM counter or disable the FTM counter. After any MCU reset, CLKS[1:0] = 0:0 so no clock source is selected.

The CLKS[1:0] bits may be read or written at any time. Disabling the FTM counter by writing 0:0 to the CLKS[1:0] bits does not affect the FTM counter value or other registers.

The fixed frequency clock is an alternative clock source for the FTM counter that allows the selection of a clock other than the system clock or an external clock. This clock input is defined by chip integration. Refer to the chip specific documentation for further information. Due to FTM hardware implementation limitations, the frequency of the fixed frequency clock must not exceed 1/2 of the system clock frequency.

The external clock passes through a synchronizer clocked by the system clock to assure that counter transitions are properly aligned to system clock transitions.Therefore, to meet Nyquist criteria considering also jitter, the frequency of the external clock source must not exceed 1/4 of the system clock frequency.

26.4.2 Prescaler

The selected counter clock source passes through a prescaler that is a 7-bit counter. The value of the prescaler is selected by the PS[2:0] bits. The following figure shows an example of the prescaler counter and FTM counter.

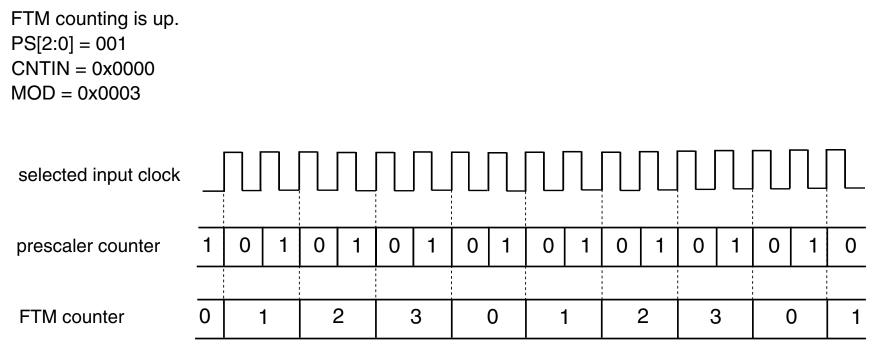


Figure 26-123. Example of the prescaler counter

26.4.3 Counter

The FTM has a 16-bit counter that is used by the channels either for input or output modes. The FTM counter clock is the selected clock divided by the prescaler.

The FTM counter has these modes of operation:

- Up counting
- Up-down counting

26.4.3.1 Up counting

Up counting is selected when:

- CPWMS = 0

CNTIN defines the starting value of the count and MOD defines the final value of the count, see the following figure. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is reloaded with the value of CNTIN.

The FTM period when using up counting is (MOD – CNTIN + 0x0001) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to CNTIN.

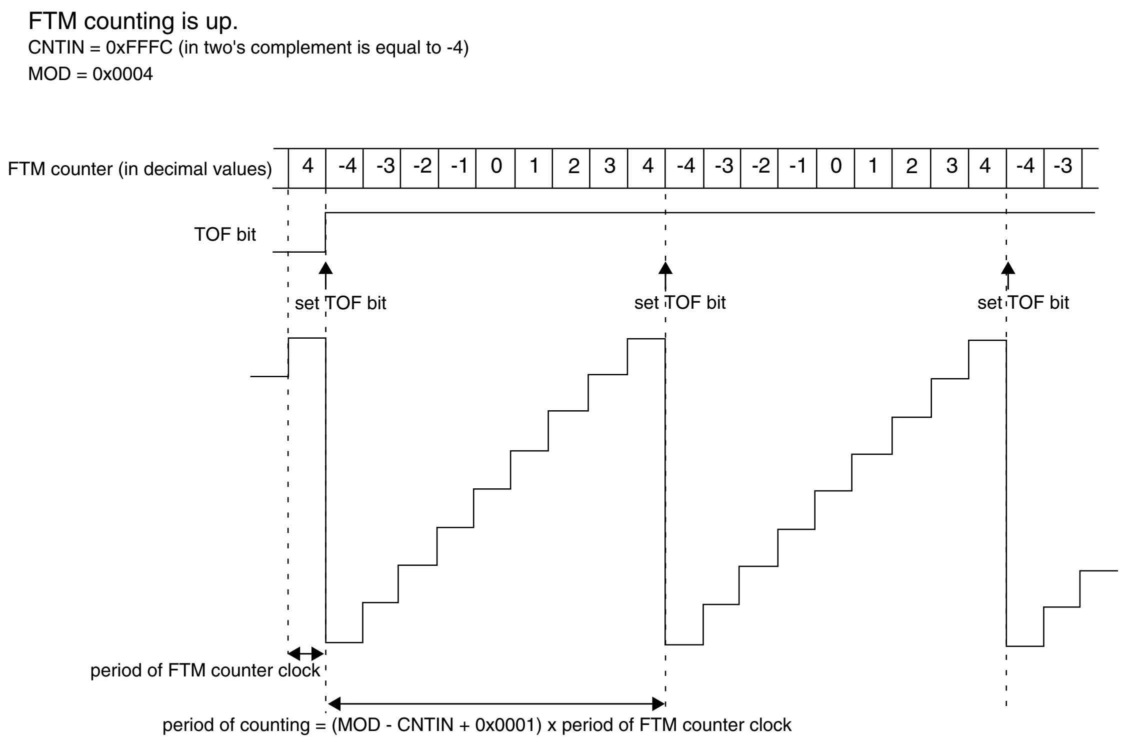

FTM counting is up.
CNTIN = 0xFFFC (in two's complement is equal to -4)
MOD = 0x0004
FTM counter (in decimal values)
4 -4 -3 -2 -1 0 1 2 3 4 -4 -3 -2 -1 0 1 2 3 4 -4 -3
TOF bit
set TOF bit
set TOF bit
set TOF bit
period of FTM counter clock
period of counting = (MOD - CNTIN + 0x0001) x period of FTM counter clock

Figure 26-124. Example of FTM up and signed counting

Table 26-179. FTM counting based on CNTIN value

When	Then
CNTIN = 0x0000	The FTM counting is equivalent to TPM up counting, that is, up and unsigned counting. See the following figure.
CNTIN[15] = 1	The initial value of the FTM counter is a negative number in two's complement, so the FTM counting is up and signed.
CNTIN[15] = 0 and CNTIN ≠ 0x0000	The initial value of the FTM counter is a positive number, so the FTM counting is up and unsigned.

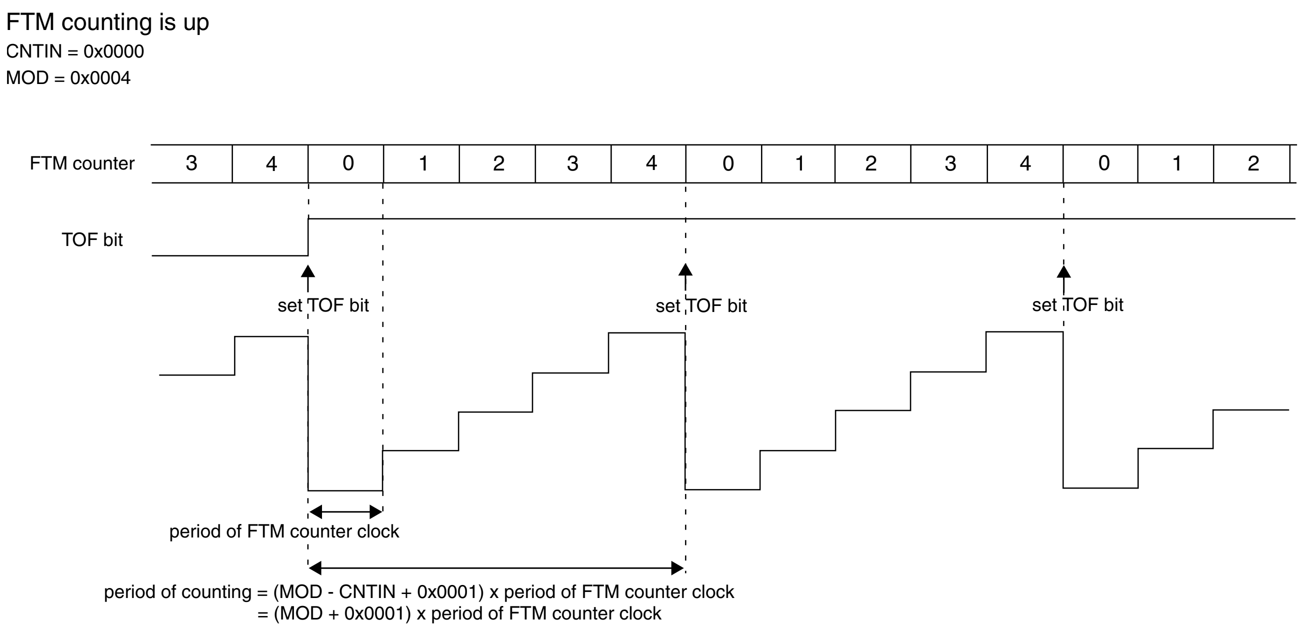

FTM counting is up
CNTIN = 0x0000
MOD = 0x0004
FTM counter
3
4
0
1
2
3
4
0
1
2
3
4
0
1
2
TOF bit
set TOF bit
set TOF bit
set TOF bit
period of FTM counter clock
period of counting = (MOD - CNTIN + 0x0001) x period of FTM counter clock
= (MOD + 0x0001) x period of FTM counter clock

Figure 26-125. Example of FTM up counting with CNTIN = 0x0000

Note

- FTM operation is only valid when the value of the CNTIN register is less than the value of the MOD register, either in the unsigned counting or signed counting. It is the responsibility of the software to ensure that the values in the CNTIN and MOD registers meet this requirement. Any values of CNTIN and MOD that do not satisfy this criteria can result in unpredictable behavior.
- MOD = CNTIN is a redundant condition. In this case, the FTM counter is always equal to MOD and the TOF bit is set in each rising edge of the FTM counter clock.
- When MOD = 0x0000, CNTIN = 0x0000, for example after reset, and FTMEN = 1, the FTM counter remains stopped at 0x0000 until a non-zero value is written into the MOD or CNTIN registers.
- Setting CNTIN to be greater than the value of MOD is not recommended as this unusual setting may make the FTM operation difficult to comprehend. However, there is no restriction on this configuration, and an example is shown in the following figure.

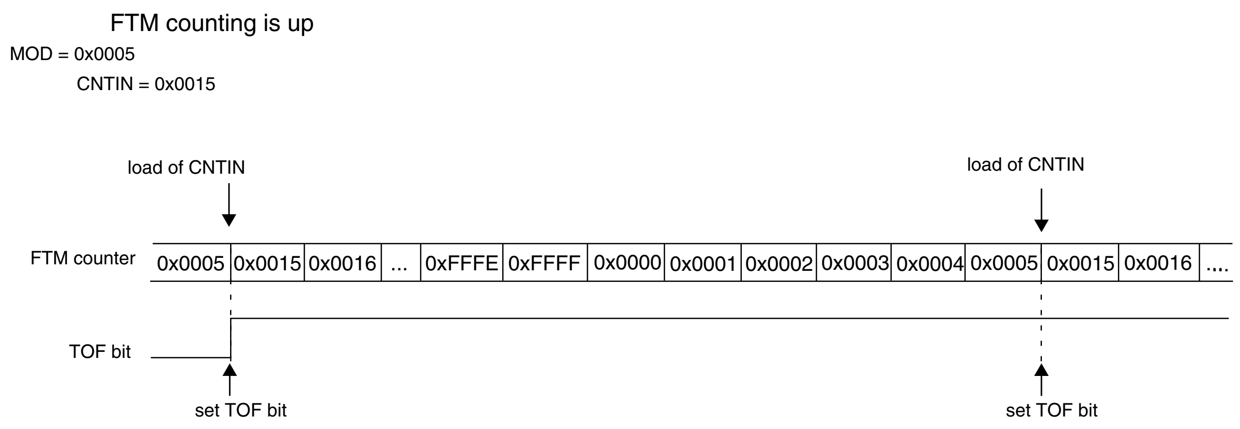


Figure 26-126. Example of up counting when the value of CNTIN is greater than the value of MOD

26.4.3.2 Up-down counting

Up-down counting is selected when:

- CPWMS = 1

CNTIN defines the starting value of the count and MOD defines the final value of the count. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is decremented until it returns to the value of CNTIN and the up-down counting restarts.

The FTM period when using up-down counting is 2 × (MOD – CNTIN) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to (MOD – 1).

If (CNTIN = 0x0000), the FTM counting is equivalent to TPM up-down counting, that is, up-down and unsigned counting. See the following figure.

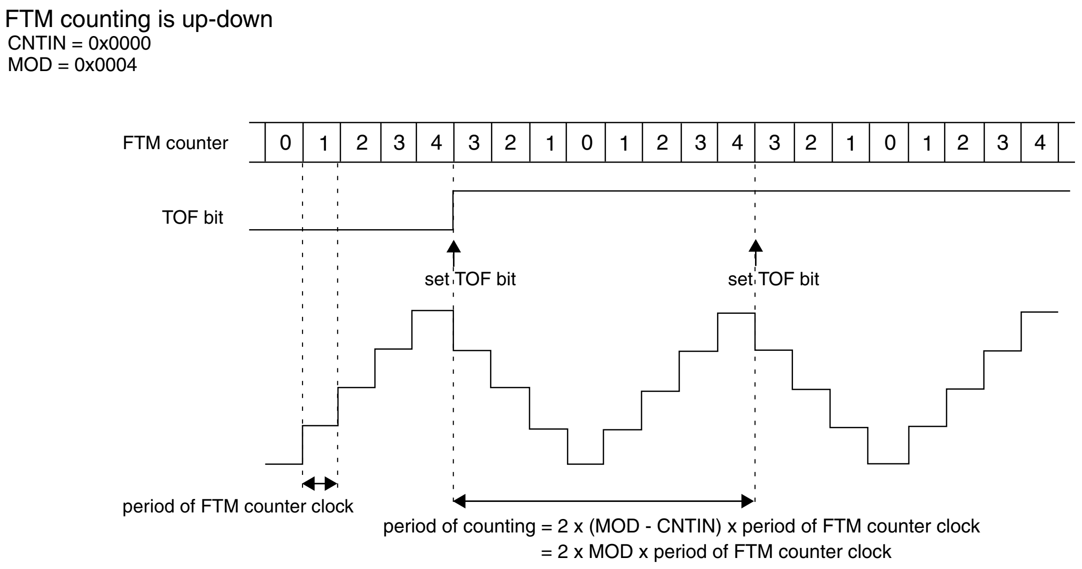

FTM counting is up-down
CNTIN = 0x0000
MOD = 0x0004
FTM counter
0 1 2 3 4 3 2 1 0 1 2 3 4 3 2 1 0 1 2 3 4
TOF bit
set TOF bit
set TOF bit
period of FTM counter clock
period of counting = 2 x (MOD - CNTIN) x period of FTM counter clock
= 2 x MOD x period of FTM counter clock

Figure 26-127. Example of up-down counting when CNTIN = 0x0000

Note

It is expected that the up-down counting be used only with CNTIN = 0x0000.

26.4.3.3 Free running counter

If (FTMEN = 0) and (MOD = 0x0000 or MOD = 0xFFFF), the FTM counter is a free running counter. In this case, the FTM counter runs free from 0x0000 through 0xFFFF and the TOF bit is set when the FTM counter changes from 0xFFFF to 0x0000. See the following figure.

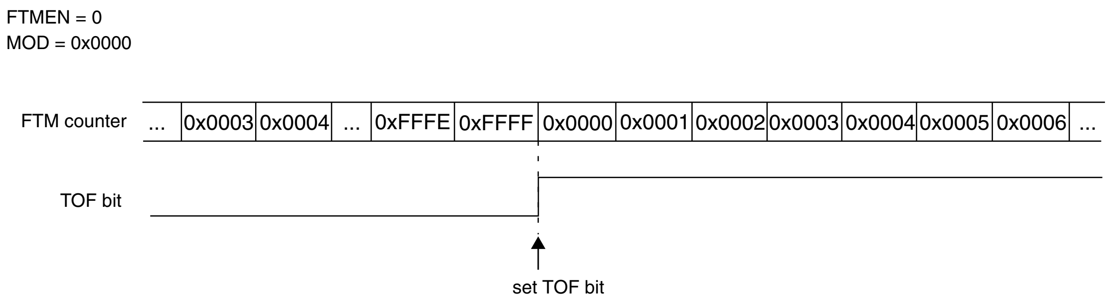

FTMEN = 0
MOD = 0x0000
FTM counter
... 0x0003 0x0004 ... 0xFFFE 0xFFFF 0x0000 0x0001 0x0002 0x0003 0x0004 0x0005 0x0006 ...
TOF bit
set TOF bit

Figure 26-128. Example when the FTM counter is free running

The FTM counter is also a free running counter when:

- FTMEN = 1
- CPWMS = 0
- CNTIN = 0x0000, and
- MOD = 0xFFFF

26.4.3.4 Counter reset

Any one of the following cases resets the FTM counter to the value in the CNTIN register and the channels output to its initial value, except for channels in Output Compare mode.

- Any write to CNT.
- FTM counter synchronization.

26.4.3.5 When the TOF bit is set

The NUMTOF[4:0] bits define the number of times that the FTM counter overflow should occur before the TOF bit to be set. If NUMTOF[4:0] = 0x00, then the TOF bit is set at each FTM counter overflow.

Initialize the FTM counter, by writing to CNT, after writing to the NUMTOF[4:0] bits to avoid confusion about when the first counter overflow will occur.

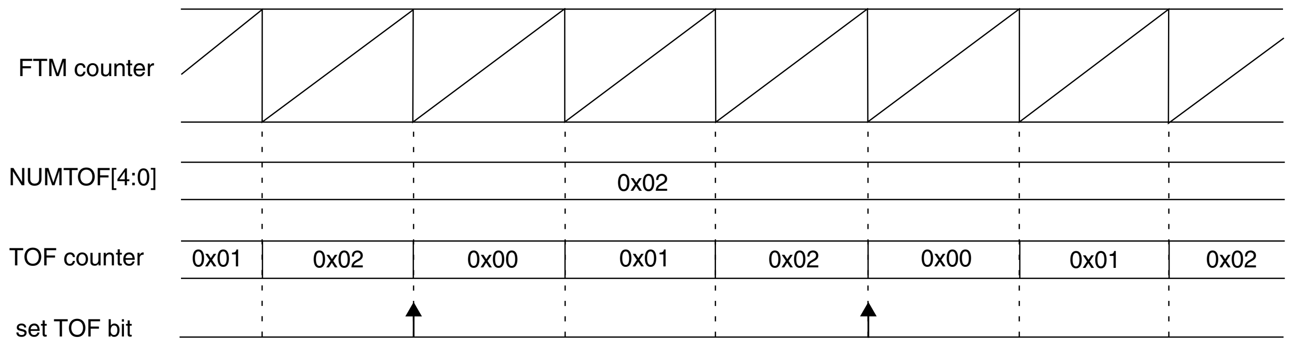


Figure 26-129. Periodic TOF when NUMTOF = 0x02

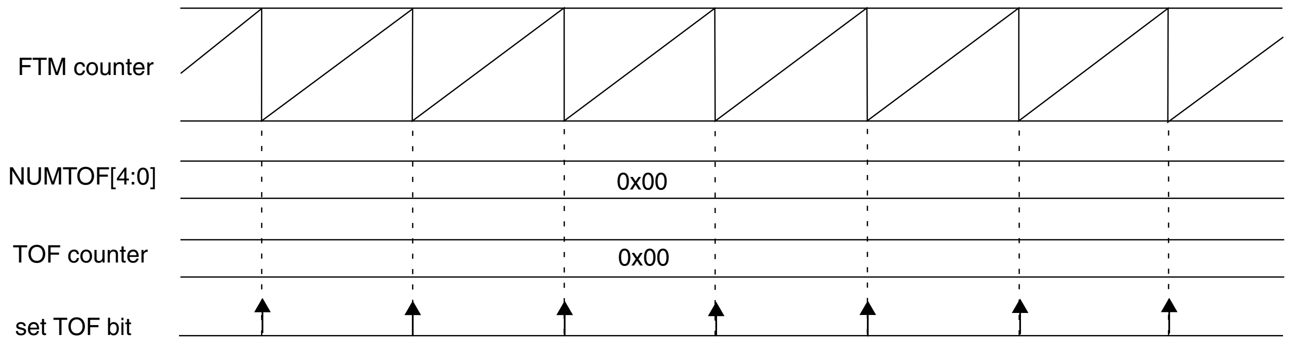


Figure 26-130. Periodic TOF when NUMTOF = 0x00

26.4.4 Input Capture mode

The Input Capture mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0
- MSnB:MSnA = 0:0, and
- ELSnB:ELSnA ≠ 0:0

When a selected edge occurs on the channel input, the current value of the FTM counter is captured into the CnV register, at the same time the CHnF bit is set and the channel interrupt is generated if enabled by CHnIE = 1. See the following figure.

When a channel is configured for input capture, the FTMxCHn pin is an edge-sensitive input. ELSnB:ELSnA control bits determine which edge, falling or rising, triggers input-capture event. Note that the maximum frequency for the channel input signal to be detected correctly is system clock divided by 4, which is required to meet Nyquist criteria for signal sampling.

Writes to the CnV register is ignored in Input Capture mode.

While in Debug mode, the input capture function works as configured. When a selected edge event occurs, the FTM counter value, which is frozen because of Debug, is captured into the CnV register and the CHnF bit is set.

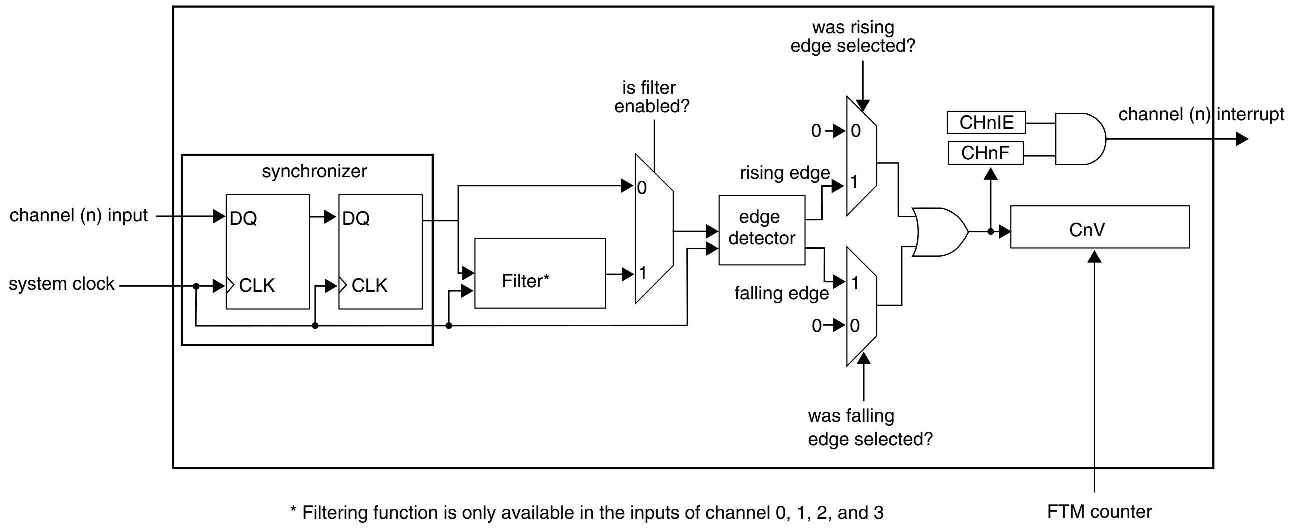


Figure 26-131. Input Capture mode

If the channel input does not have a filter enabled, then the input signal is always delayed 3 rising edges of the system clock, that is, two rising edges to the synchronizer plus one more rising edge to the edge detector. In other words, the CHnF bit is set on the third rising edge of the system clock after a valid edge occurs on the channel input.

Note

The Input Capture mode must be used only with CNTIN = 0x0000.

26.4.4.1 Filter for Input Capture mode

The filter function is only available on channels 0, 1, 2, and 3.

First, the input signal is synchronized by the system clock. Following synchronization, the input signal enters the filter block. See the following figure.

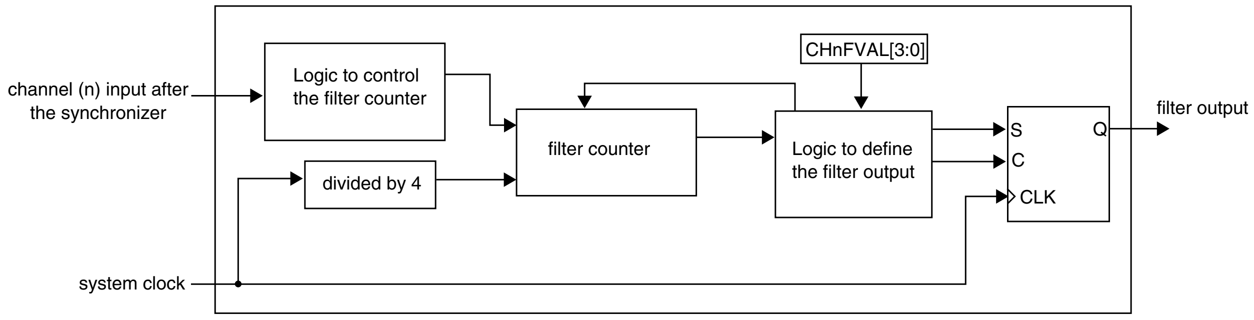

channel (n) input after the synchronizer
Logic to control the filter counter
divided by 4
system clock
filter counter
CHnFVAL[3:0]
Logic to define the filter output
S
C
CLK
Q
filter output

Figure 26-132. Channel input filter

When there is a state change in the input signal, the counter is reset and starts counting up. As long as the new state is stable on the input, the counter continues to increment. When the counter is equal to CHnFVAL[3:0], the state change of the input signal is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the input signal before it can be validated, the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by CHnFVAL[3:0] (× 4 system clocks) is regarded as a glitch and is not passed on to the edge detector. A timing diagram of the input filter is shown in the following figure.

The filter function is disabled when CHnFVAL[3:0] bits are zero. In this case, the input signal is delayed 3 rising edges of the system clock. If (CHnFVAL[3:0] ≠ 0000), then the input signal is delayed by the minimum pulse width (CHnFVAL[3:0] × 4 system clocks) plus a further 4 rising edges of the system clock: two rising edges to the synchronizer, one rising edge to the filter output, plus one more to the edge detector. In other words, CHnF is set (4 + 4 × CHnFVAL[3:0]) system clock periods after a valid edge occurs on the channel input.

The clock for the counter in the channel input filter is the system clock divided by 4.

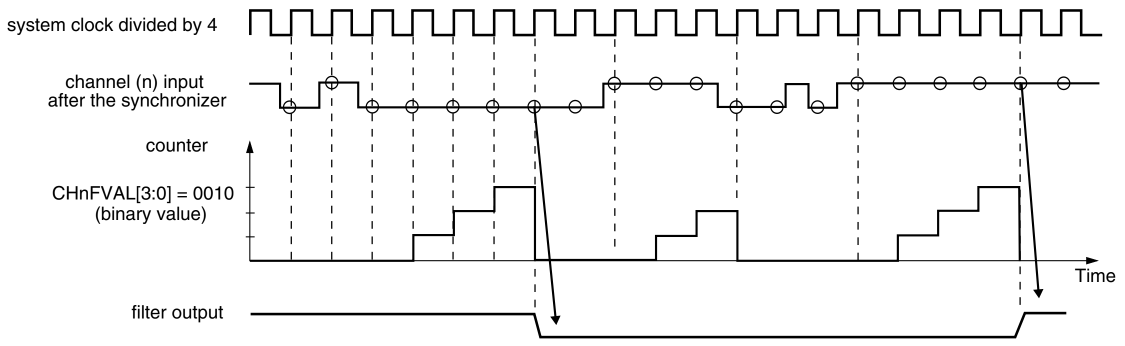

system clock divided by 4
channel (n) input after the synchronizer
counter
CHnFVAL[3:0] = 0010 (binary value)
Time
filter output

Figure 26-133. Channel input filter example

26.4.5 Output Compare mode

The Output Compare mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB:MSnA = 0:1

In Output Compare mode, the FTM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CnV register of an output compare channel, the channel (n) output can be set, cleared, or toggled.

When a channel is initially configured to Toggle mode, the previous value of the channel output is held until the first output compare event occurs.

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV).

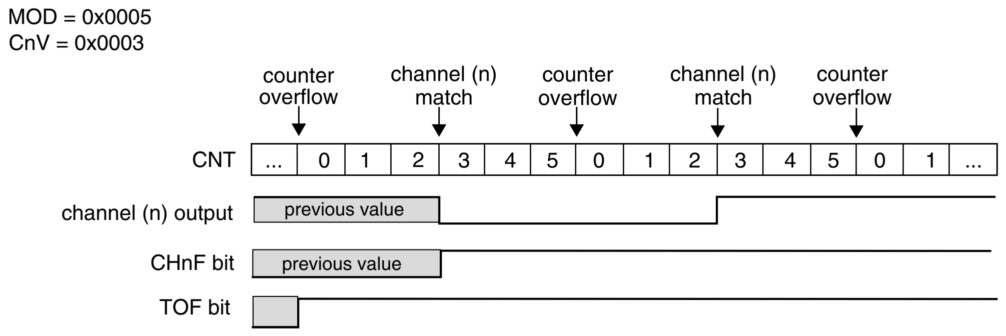


Figure 26-134. Example of the Output Compare mode when the match toggles the channel output

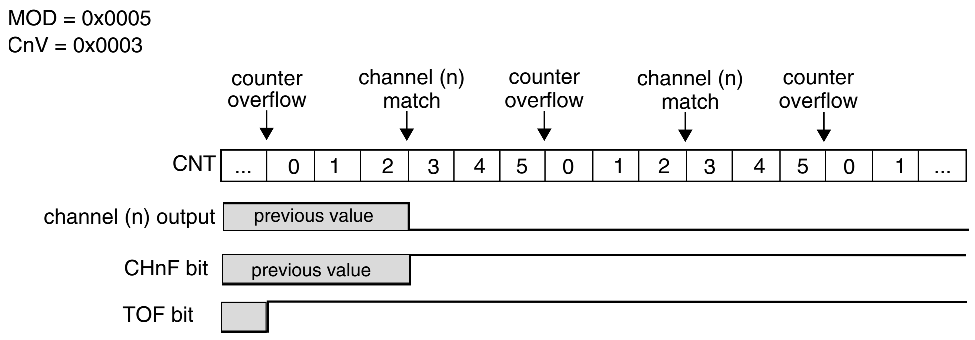


Figure 26-135. Example of the Output Compare mode when the match clears the channel output

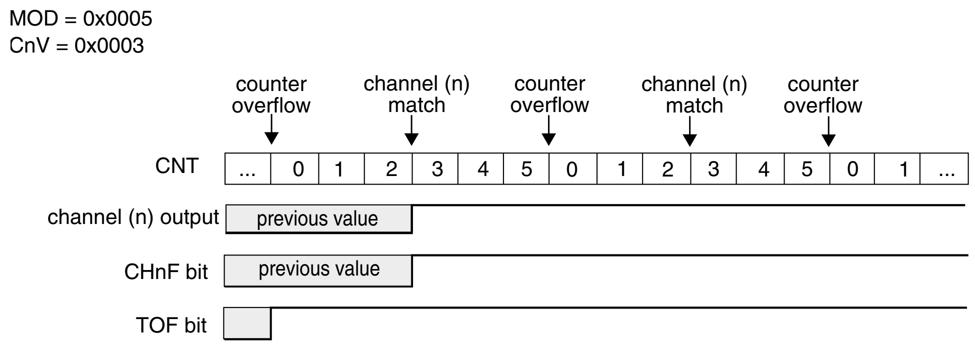


Figure 26-136. Example of the Output Compare mode when the match sets the channel output

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not modified and controlled by FTM.

Note

The Output Compare mode must be used only with CNTIN = 0x0000.

26.4.6 Edge-Aligned PWM (EPWM) mode

The Edge-Aligned mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB = 1

The EPWM period is determined by (MOD – CNTIN + 0x0001) and the pulse width (duty cycle) is determined by (CnV – CNTIN).

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV), that is, at the end of the pulse width.

This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an FTM.

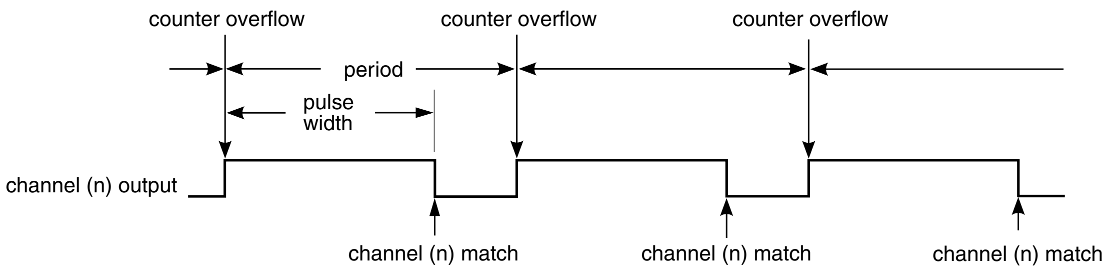


Figure 26-137. EPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced low at the channel (n) match (FTM counter = CnV). See the following figure.

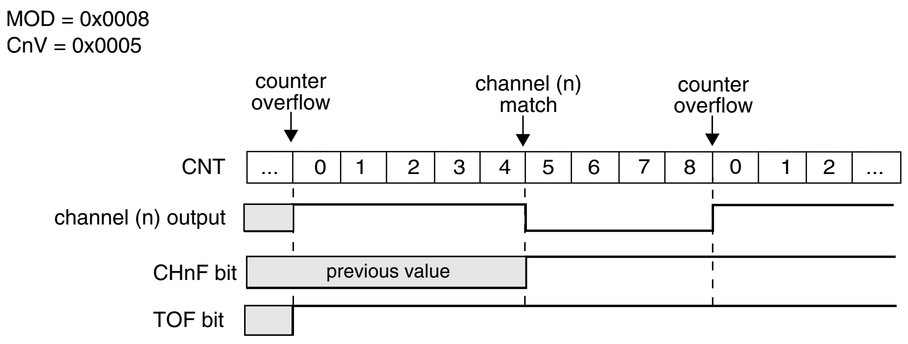


Figure 26-138. EPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced high at the channel (n) match (FTM counter = CnV). See the following figure.

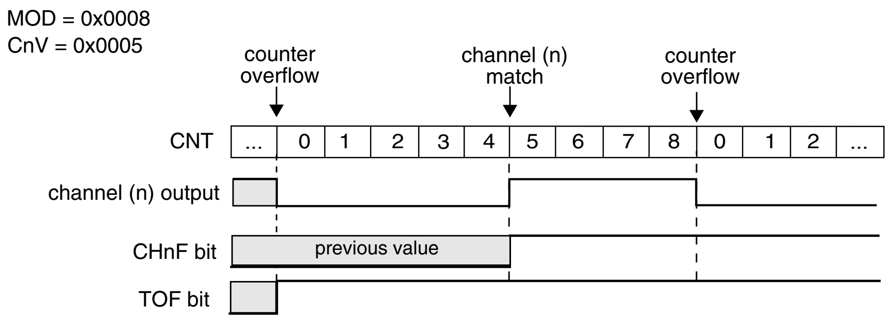


Figure 26-139. EPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000), then the channel (n) output is a 0% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match. If (CnV > MOD), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match. Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

Note

The EPWM mode must be used only with CNTIN = 0x0000.

26.4.7 Center-Aligned PWM (CPWM) mode

The Center-Aligned mode is selected when:

- DECAPEN = 0
- COMBINE = 0, and
- CPWMS = 1

The CPWM pulse width (duty cycle) is determined by 2 × (CnV – CNTIN) and the period is determined by 2 × (MOD – CNTIN). See the following figure. MOD must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

In the CPWM mode, the FTM counter counts up until it reaches MOD and then counts down until it reaches CNTIN.

The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = CnV) when the FTM counting is down (at the begin of the pulse width) and when the FTM counting is up (at the end of the pulse width).

This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value of CNTIN.

The other channel modes are not compatible with the up-down counter (CPWMS = 1). Therefore, all FTM channels must be used in CPWM mode when (CPWMS = 1).

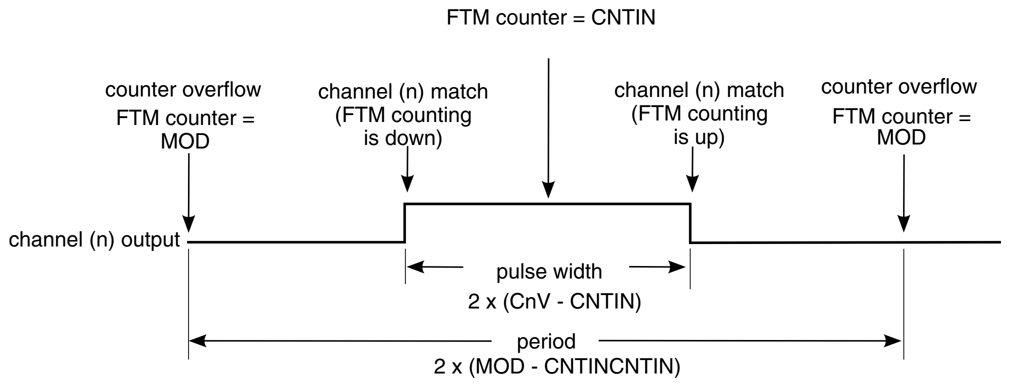


Figure 26-140. CPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the FTM counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the channel (n) match (FTM counter = CnV) when counting down, and it is forced low at the channel (n) match when counting up. See the following figure.

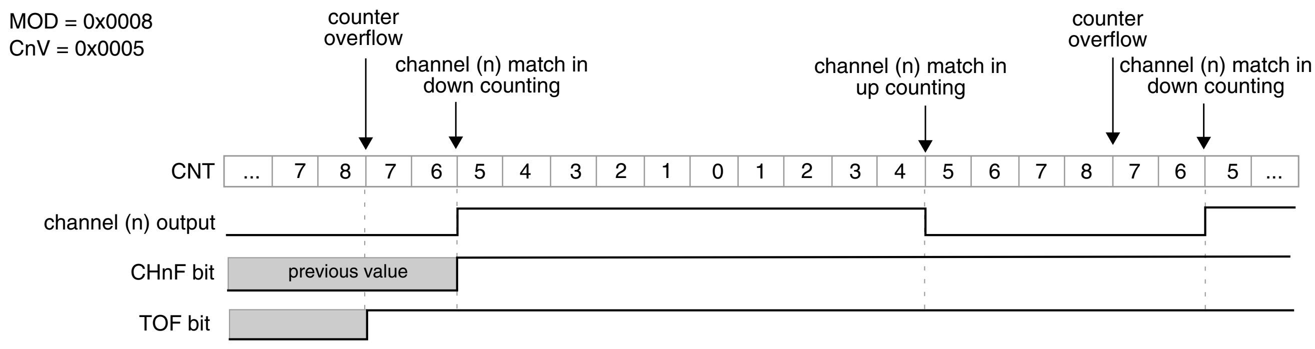


Figure 26-141. CPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the channel (n) match (FTM counter = CnV) when counting down, and it is forced high at the channel (n) match when counting up. See the following figure.

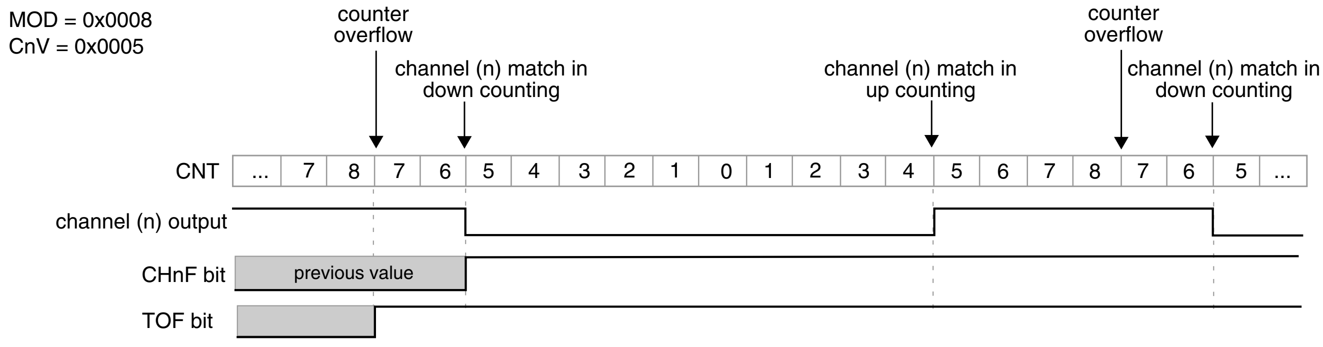


Figure 26-142. CPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000) or CnV is a negative value, that is (CnV[15] = 1), then the channel (n) output is a 0% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match.

If CnV is a positive value, that is (CnV[15] = 0), (CnV ≥ MOD), and (MOD ≠ 0x0000), then the channel (n) output is a 100% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match. This implies that the usable range of periods set by MOD is 0x0001 through 0x7FFE, 0x7FFF if you do not need to generate a 100% duty cycle CPWM signal. This is not a significant limitation because the resulting period is much longer than required for normal applications.

The CPWM mode must not be used when the FTM counter is a free running counter.

Note

The CPWM mode must be used only with CNTIN = 0x0000.

26.4.8 Combine mode

The Combine mode is selected when:

- FTMEN = 1
- DECAPEN = 0
- COMBINE = 1, and
- CPWMS = 0

In Combine mode, an even channel (n) and adjacent odd channel (n+1) are combined to generate a PWM signal in the channel (n) output.

In the Combine mode, the PWM period is determined by (MOD − CNTIN + 0x0001) and the PWM pulse width (duty cycle) is determined by (|C(n+1)V − C(n)V|).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = C(n)V). The CH(n+1)F bit is set and the channel (n +1) interrupt is generated, if CH(n+1)IE = 1, at the channel (n+1) match (FTM counter = C(n+1)V).

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n +1)V). It is forced high at the channel (n) match (FTM counter = C(n)V). See the following figure.

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced high at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n+1)V). It is forced low at the channel (n) match (FTM counter = C(n)V). See the following figure.

In Combine mode, the ELS(n+1)B and ELS(n+1)A bits are not used in the generation of the channels (n) and (n+1) output. However, if (ELSnB:ELSnA = 0:0) then the channel (n) output is not controlled by FTM, and if (ELS(n+1)B:ELS(n+1)A = 0:0) then the channel (n+1) output is not controlled by FTM.

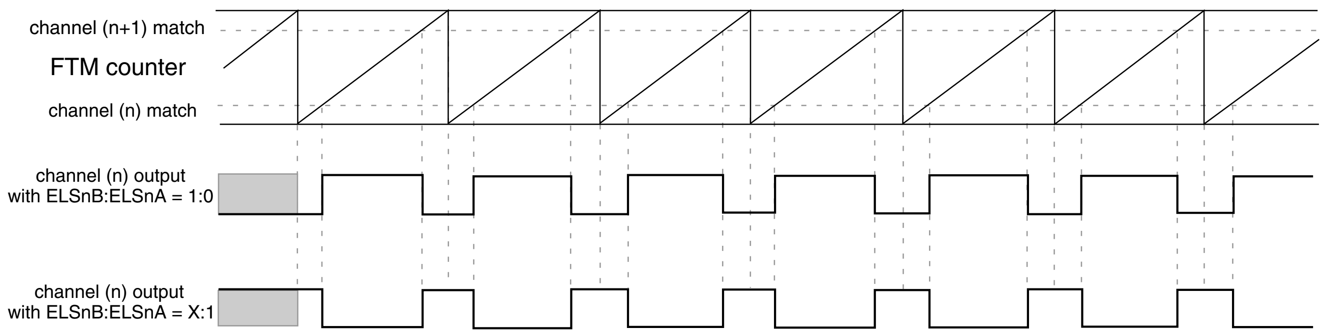


Figure 26-143. Combine mode

The following figures illustrate the PWM signals generation using Combine mode.

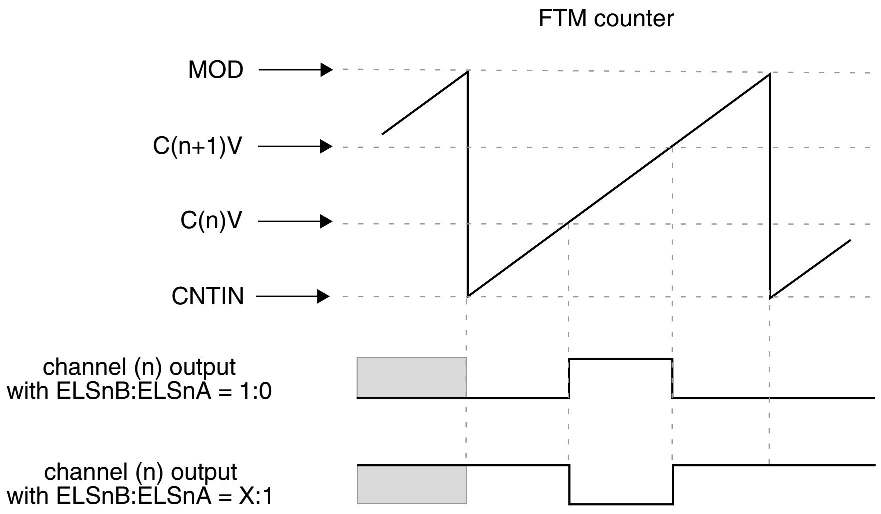


Figure 26-144. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V < C(n+1)V)

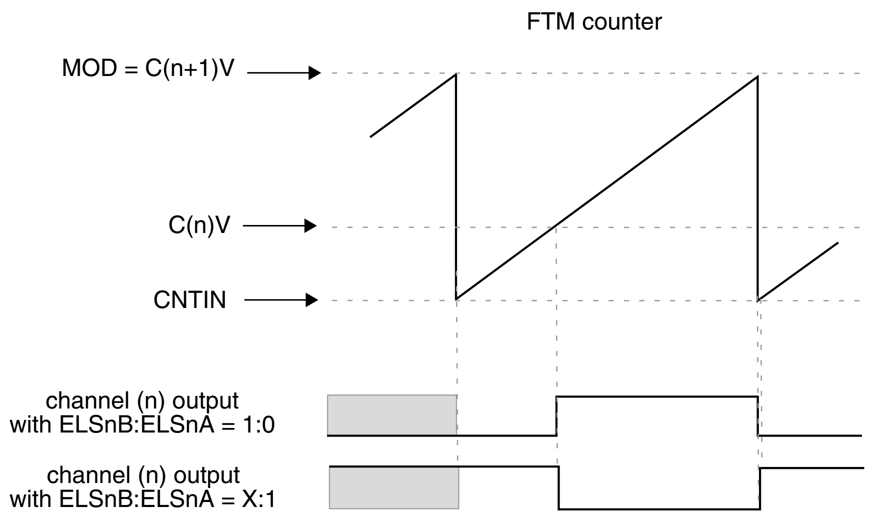

FTM counter
MOD = C(n+1)V
C(n)V
CNTIN
channel (n) output with ELSnB:ELSnA = 1:0
channel (n) output with ELSnB:ELSnA = X:1

Figure 26-145. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n+1)V = MOD)

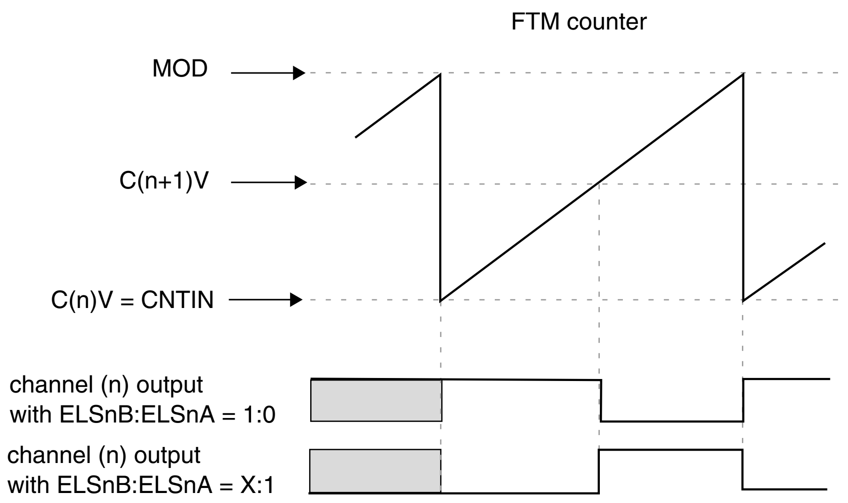

FTM counter
MOD
C(n+1)V
C(n)V = CNTIN
channel (n) output with ELSnB:ELSnA = 1:0
channel (n) output with ELSnB:ELSnA = X:1

Figure 26-146. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD)

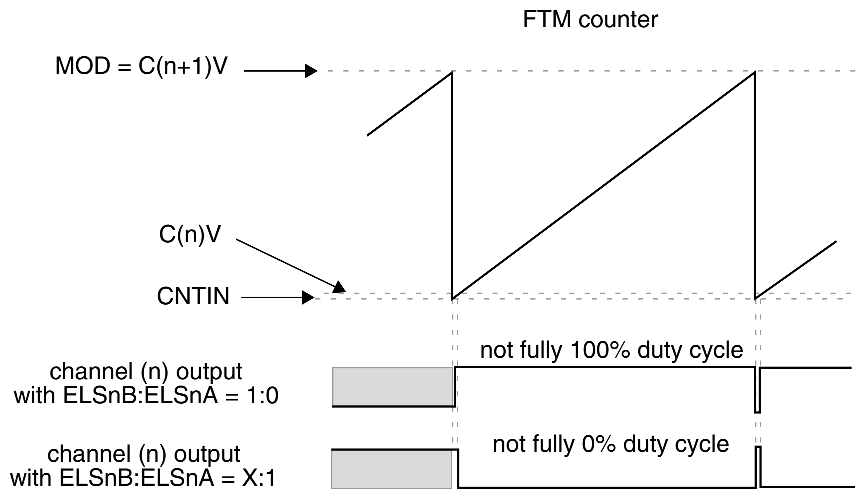

FTM counter
MOD = C(n+1)V
C(n)V
CNTIN
not fully 100% duty cycle
channel (n) output with ELSnB:ELSnA = 1:0
not fully 0% duty cycle
channel (n) output with ELSnB:ELSnA = X:1

Figure 26-147. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n)V is Almost Equal to CNTIN) and (C(n+1)V = MOD)

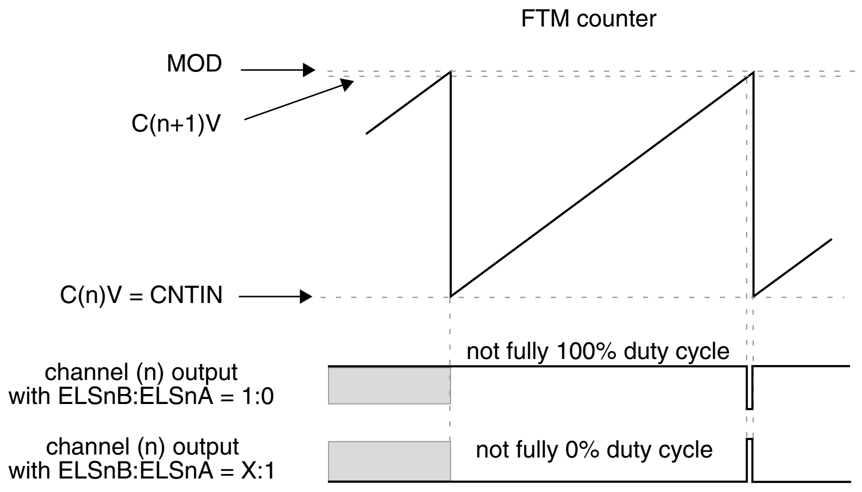


Figure 26-148. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD) and (C(n+1)V is Almost Equal to MOD)

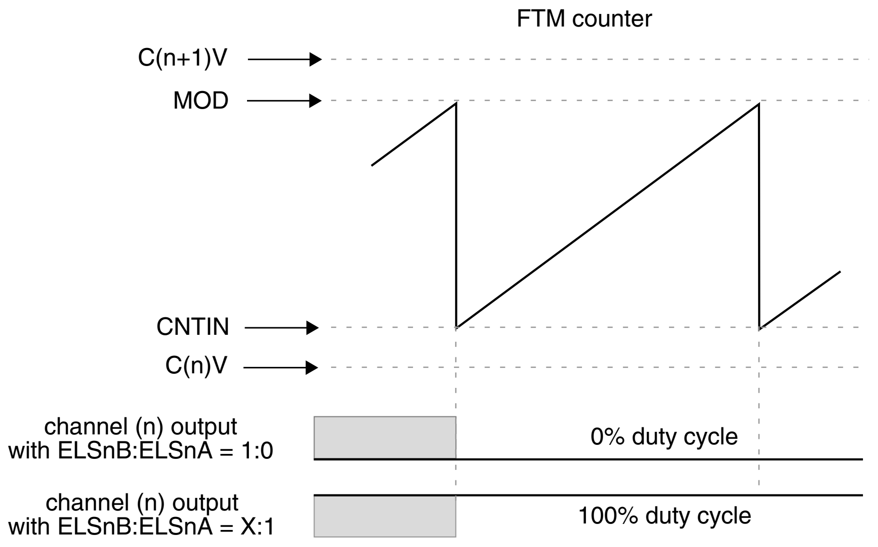


Figure 26-149. Channel (n) output if C(n)V and C(n+1)V are not between CNTIN and MOD

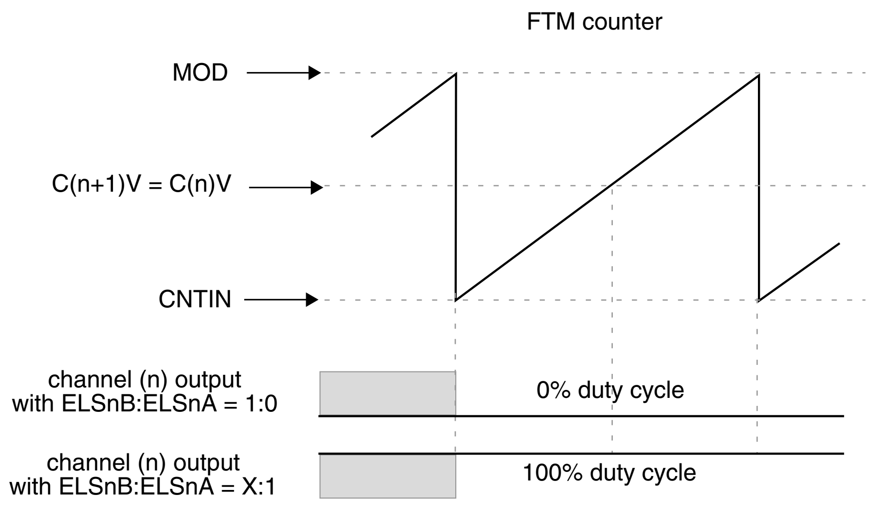


Figure 26-150. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V = C(n+1)V)

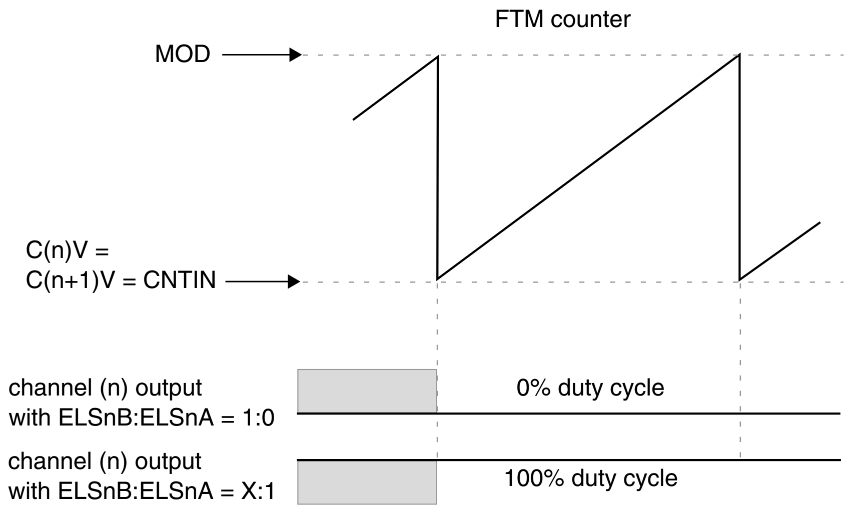


Figure 26-151. Channel (n) output if (C(n)V = C(n+1)V = CNTIN)

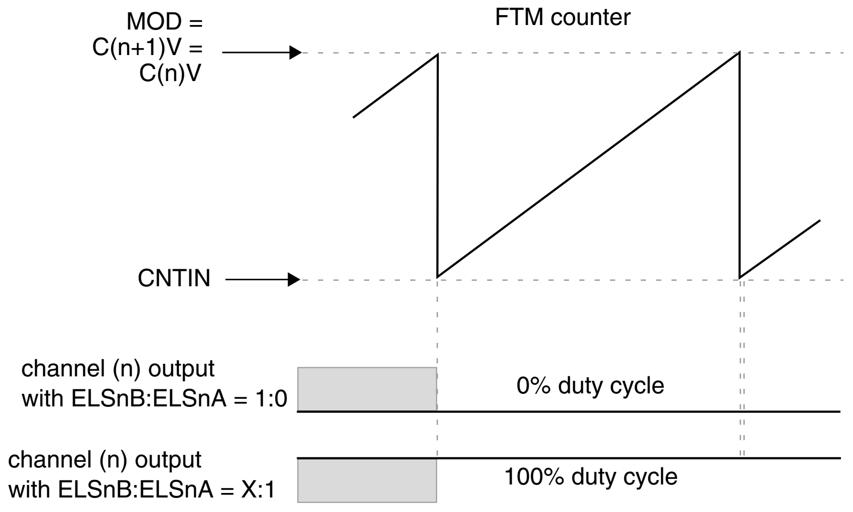


Figure 26-152. Channel (n) output if (C(n)V = C(n+1)V = MOD)

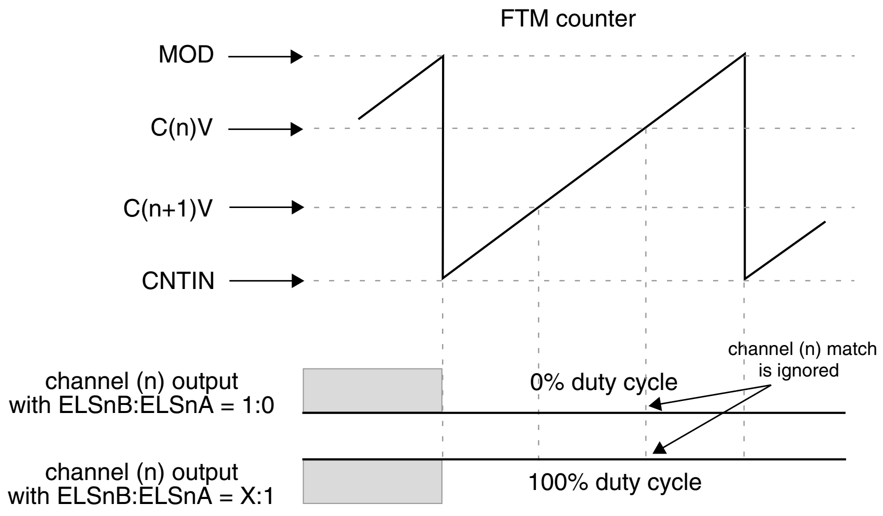


Figure 26-153. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V > C(n+1)V)

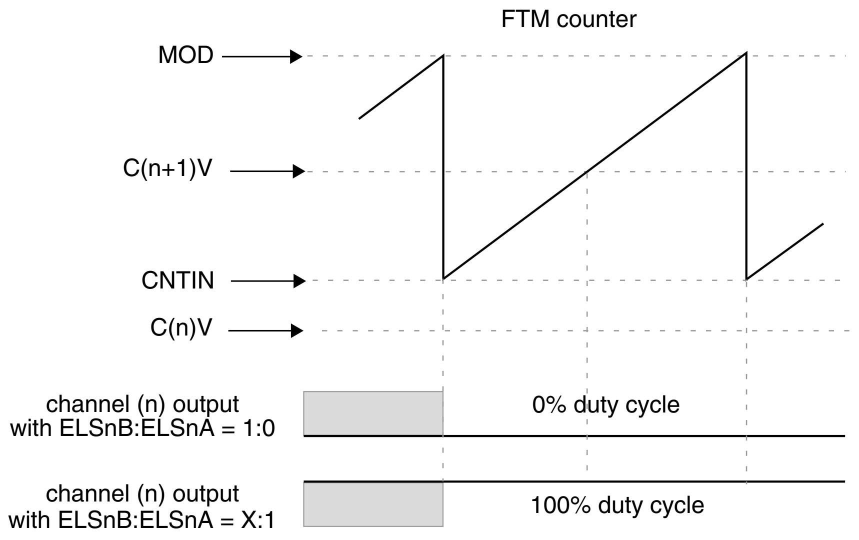


Figure 26-154. Channel (n) output if (C(n)V < CNTIN) and (CNTIN < C(n+1)V < MOD)

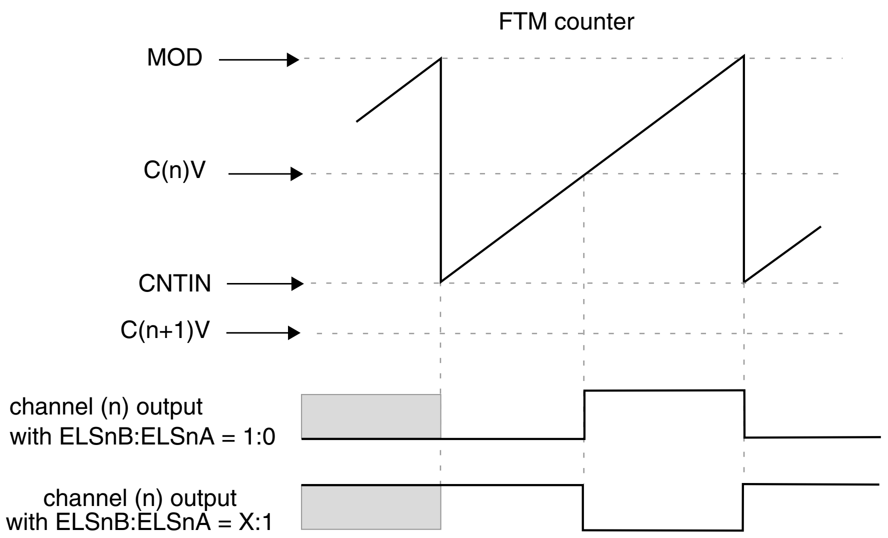


Figure 26-155. Channel (n) output if (C(n+1)V < CNTIN) and (CNTIN < C(n)V < MOD)

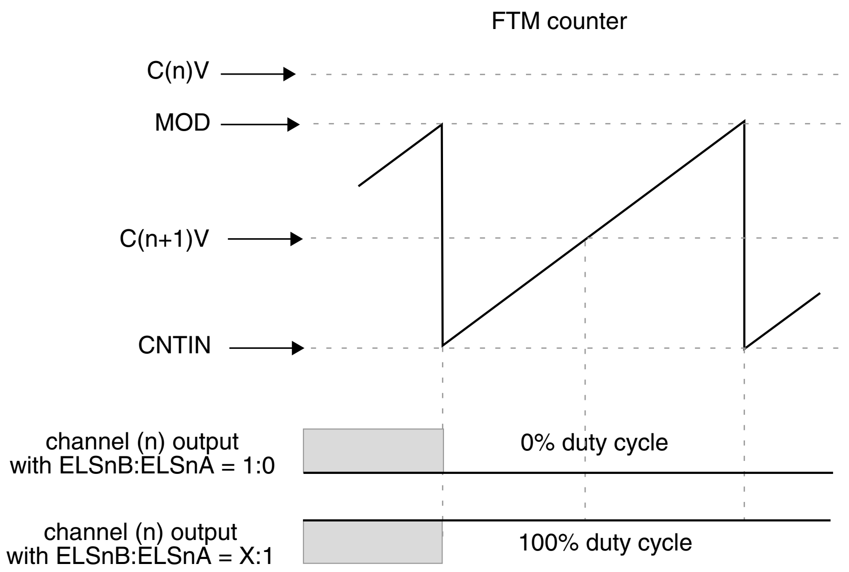


Figure 26-156. Channel (n) output if (C(n)V > MOD) and (CNTIN < C(n+1)V < MOD)

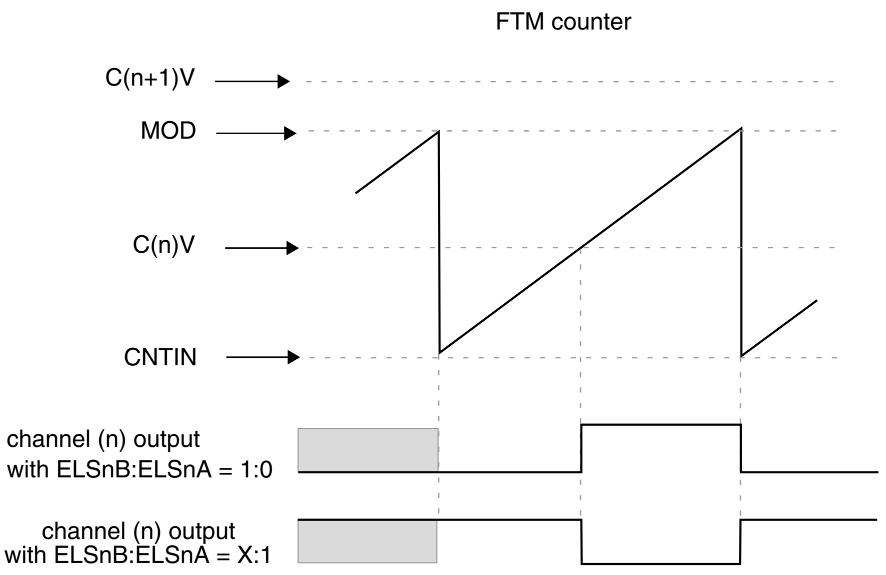


Figure 26-157. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V < MOD)

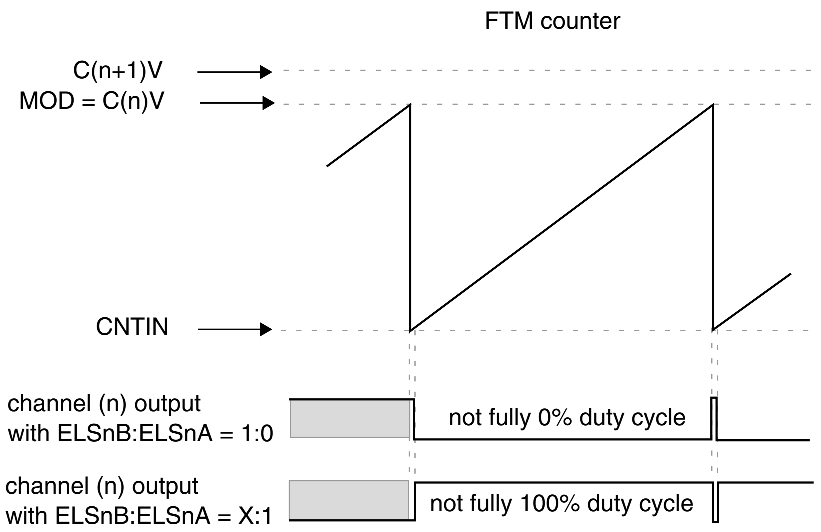


Figure 26-158. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V = MOD)

26.4.8.1 Asymmetrical PWM

In Combine mode, the control of the PWM signal first edge, when the channel (n) match occurs, that is, FTM counter = C(n)V, is independent of the control of the PWM signal second edge, when the channel (n+1) match occurs, that is, FTM counter = C(n+1)V. So, Combine mode allows the generation of asymmetrical PWM signals.

26.4.9 Complementary mode

The Complementary mode is selected when:

- FTMEN = 1
- DECAPEN = 0
- COMBINE = 1
- CPWMS = 0, and
- COMP = 1

In Complementary mode, the channel (n+1) output is the inverse of the channel (n) output.

So, the channel (n+1) output is the same as the channel (n) output when:

- FTMEN = 1
- DECAPEN = 0
- COMBINE = 1
- CPWMS = 0, and
- COMP = 0

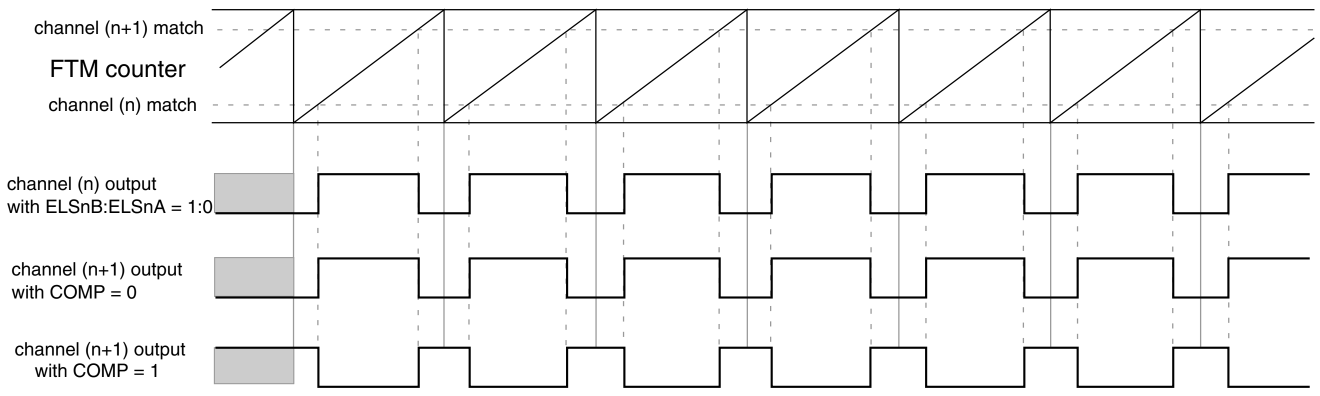

channel (n+1) match
FTM counter
channel (n) match
channel (n) output with ELSnB:ELSnA = 1:0
channel (n+1) output with COMP = 0
channel (n+1) output with COMP = 1

Figure 26-159. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = 1:0)

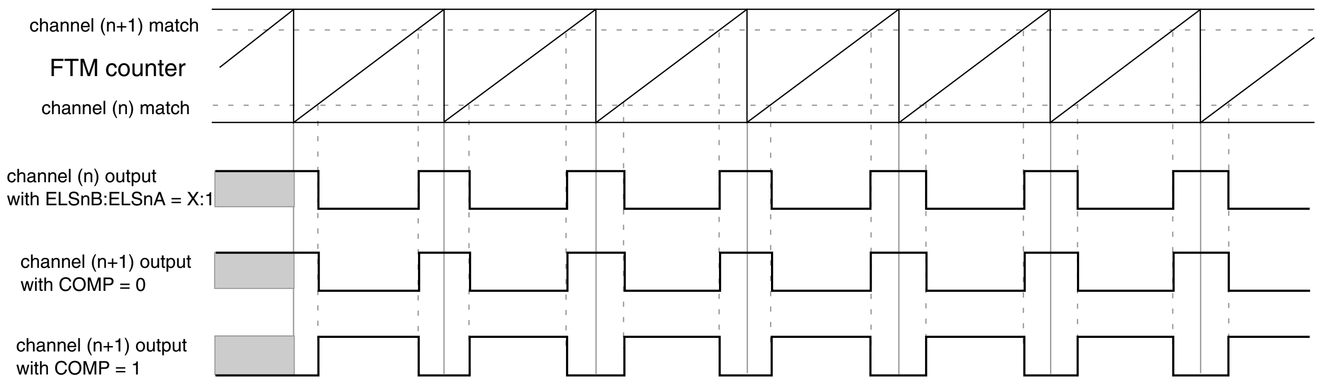

channel (n+1) match
FTM counter
channel (n) match
channel (n) output with ELSnB:ELSnA = X:1
channel (n+1) output with COMP = 0
channel (n+1) output with COMP = 1

Figure 26-160. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = X:1)

26.4.10 Registers updated from write buffers

26.4.10.1 CNTIN register update

The following table describes when CNTIN register is updated:

Table 26-180. CNTIN register update

When	Then CNTIN register is updated
CLKS[1:0] = 0:0	When CNTIN register is written, independent of FTMEN bit.
• FTMEN = 0, or • CNTINC = 0	At the next system clock after CNTIN was written.
• FTMEN = 1, • SYNCMODE = 1, and • CNTINC = 1	By the CNTIN register synchronization.

26.4.10.2 MOD register update

The following table describes when MOD register is updated:

Table 26-181. MOD register update

When	Then MOD register is updated
CLKS[1:0] = 0:0	When MOD register is written, independent of FTMEN bit.
• CLKS[1:0] ≠ 0:0, and • FTMEN = 0	According to the CPWMS bit, that is: • If the selected mode is not CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000. • If the selected mode is CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to (MOD – 0x0001).
• CLKS[1:0] ≠ 0:0, and • FTMEN = 1	By the MOD register synchronization.

26.4.10.3 CnV register update

The following table describes when CnV register is updated:

Table 26-182. CnV register update

When	Then CnV register is updated
CLKS[1:0] = 0:0	When CnV register is written, independent of FTMEN bit.
• CLKS[1:0] ≠ 0:0, and • FTMEN = 0	According to the selected mode, that is:

Table continues on the next page...

Table 26-182. CnV register update (continued)

When	Then CnV register is updated
	• If the selected mode is Output Compare, then CnV register is updated on the next FTM counter change, end of the prescaler counting, after CnV register was written. • If the selected mode is EPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000. • If the selected mode is CPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to (MOD – 0x0001).
• CLKS[1:0] ≠ 0:0, and • FTMEN = 1	According to the selected mode, that is: • If the selected mode is output compare then CnV register is updated according to the SYNCEN bit. If (SYNCEN = 0) then CnV register is updated after CnV register was written at the next change of the FTM counter, the end of the prescaler counting. If (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization. • If the selected mode is not output compare and (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization.

26.4.11 PWM synchronization

The PWM synchronization provides an opportunity to update the MOD, CNTIN, CnV, OUTMASK, INVCTRL and SWOCTRL registers with their buffered value and force the FTM counter to the CNTIN register value.

Note

- The PWM synchronization must be used only in Combine mode.
- The legacy PWM synchronization (SYNCMODE = 0) is a subset of the enhanced PWM synchronization (SYNCMODE = 1). Thus, only the enhanced PWM synchronization must be used.

26.4.11.1 Hardware trigger

Three hardware trigger signal inputs of the FTM module are enabled when TRIGn = 1, where n = 0, 1 or 2 corresponding to each one of the input signals, respectively. The hardware trigger input n is synchronized by the system clock. The PWM synchronization with hardware trigger is initiated when a rising edge is detected at the enabled hardware trigger inputs.

If (HWTRIGMODE = 0) then the TRIGn bit is cleared when 0 is written to it or when the trigger n event is detected.

In this case, if two or more hardware triggers are enabled (for example, TRIG0 and TRIG1 = 1) and only trigger 1 event occurs, then only TRIG1 bit is cleared. If a trigger n event occurs together with a write setting TRIGn bit, then the synchronization is initiated, but TRIGn bit remains set due to the write operation.

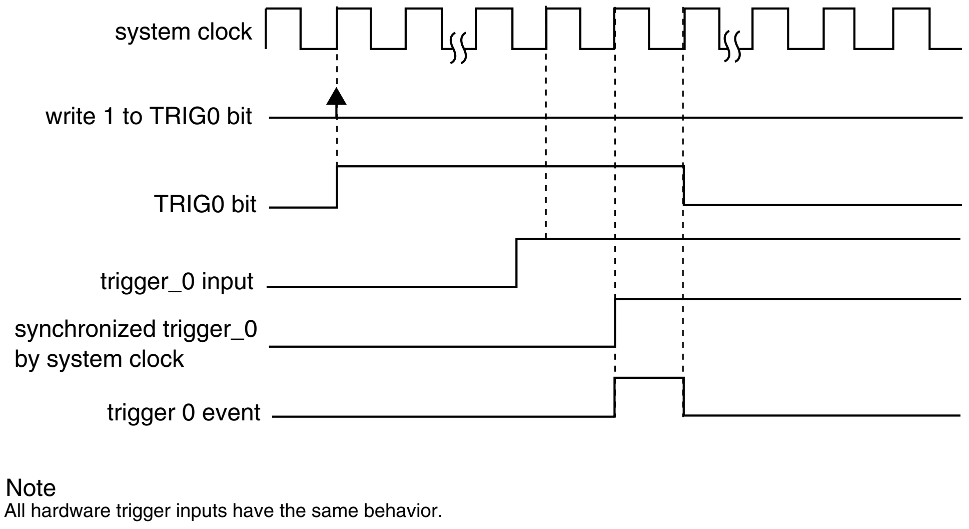

system clock
write 1 to TRIG0 bit
TRIG0 bit
trigger_0 input
synchronized trigger_0 by system clock
trigger 0 event
Note
All hardware trigger inputs have the same behavior.

Figure 26-161. Hardware trigger event with HWTRIGMODE = 0

If HWTRIGMODE = 1, then the TRIGn bit is only cleared when 0 is written to it.

NOTE

The HWTRIGMODE bit must be 1 only with enhanced PWM synchronization (SYNCMODE = 1).

26.4.11.2 Software trigger

A software trigger event occurs when 1 is written to the SYNC[SWSYNC] bit. The SWSYNC bit is cleared when 0 is written to it or when the PWM synchronization, initiated by the software event, is completed.

If another software trigger event occurs (by writing another 1 to the SWSYNC bit) at the same time the PWM synchronization initiated by the previous software trigger event is ending, a new PWM synchronization is started and the SWSYNC bit remains equal to 1.

If SYNCMODE = 0 then the SWSYNC bit is also cleared by FTM according to PWMSYNC and REINIT bits. In this case if (PWMSYNC = 1) or (PWMSYNC = 0 and REINIT = 0) then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see Boundary cycle and loading points and the following figure. If (PWMSYNC = 0) and (REINIT = 1) then SWSYNC bit is cleared when the software trigger event occurs.

If SYNCMODE = 1 then the SWSYNC bit is also cleared by FTM according to the SWRSTCNT bit. If SWRSTCNT = 0 then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see the following figure. If SWRSTCNT = 1 then SWSYNC bit is cleared when the software trigger event occurs.

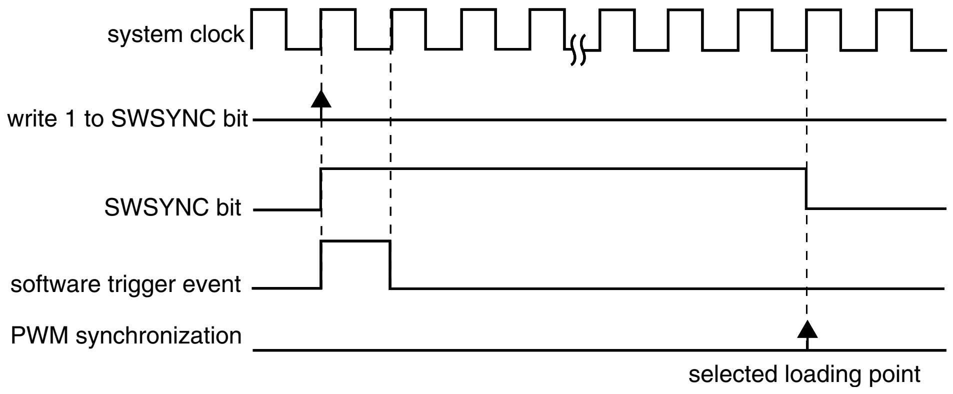


Figure 26-162. Software trigger event

26.4.11.3 Boundary cycle and loading points

The boundary cycle definition is important for the loading points for the registers MOD, CNTIN, and C(n)V.

In Up counting mode, the boundary cycle is defined as when the counter wraps to its initial value (CNTIN). If in Up-down counting mode, then the boundary cycle is defined as when the counter turns from down to up counting and when from up to down counting.

The following figure shows the boundary cycles and the loading points for the registers. In the Up Counting mode, the loading points are enabled if one of CNTMIN or CTMAX bits are 1. In the Up-Down Counting mode, the loading points are selected by CNTMIN and CNTMAX bits, as indicated in the figure. These loading points are safe places for register updates thus allowing a smooth transitions in PWM waveform generation.

For both counting modes, if neither CNTMIN nor CNTMAX are 1, then the boundary cycles are not used as loading points for registers updates. See the register synchronization descriptions in the following sections for details.

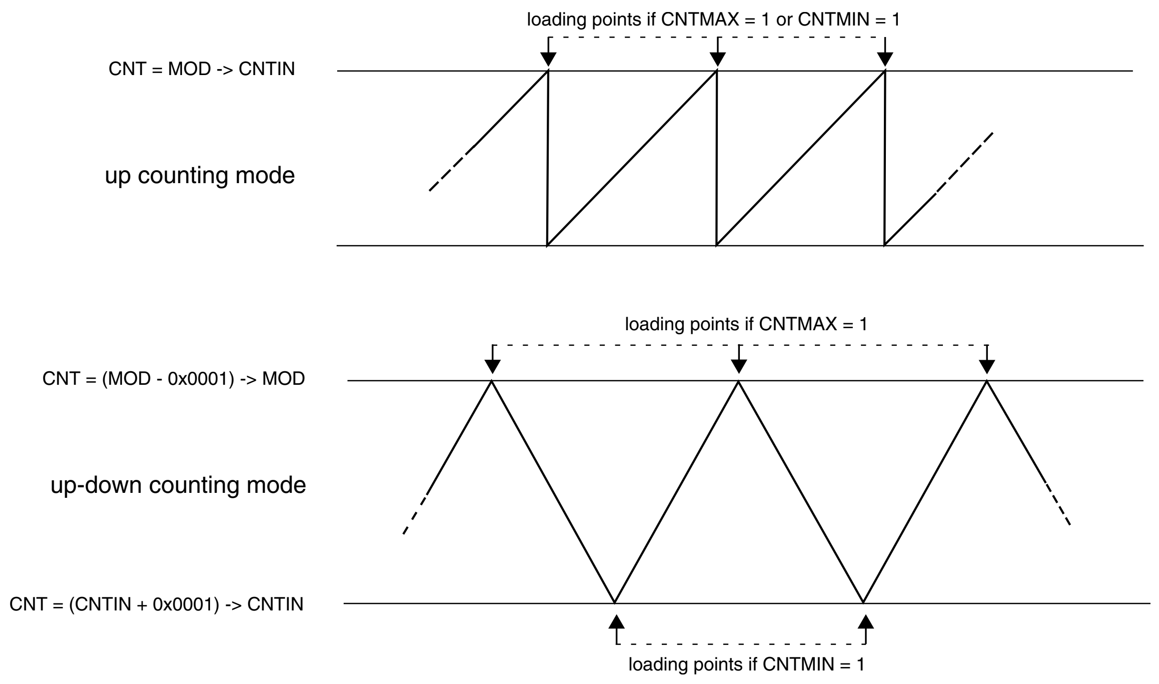

loading points if CNTMAX = 1 or CNTMIN = 1
CNT = MOD -> CNTIN
up counting mode
loading points if CNTMAX = 1
CNT = (MOD - 0x0001) -> MOD
up-down counting mode
CNT = (CNTIN + 0x0001) -> CNTIN
loading points if CNTMIN = 1

Figure 26-163. Boundary cycles and loading points

26.4.11.4 MOD register synchronization

The MOD register synchronization updates the MOD register with its buffer value. This synchronization is enabled if (FTMEN = 1).

The MOD register synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, it is expected that the MOD register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the MOD register synchronization depends on SWWRBUF, SWRSTCNT, HWWRBUF, and HWRSTCNT bits according to this flowchart:

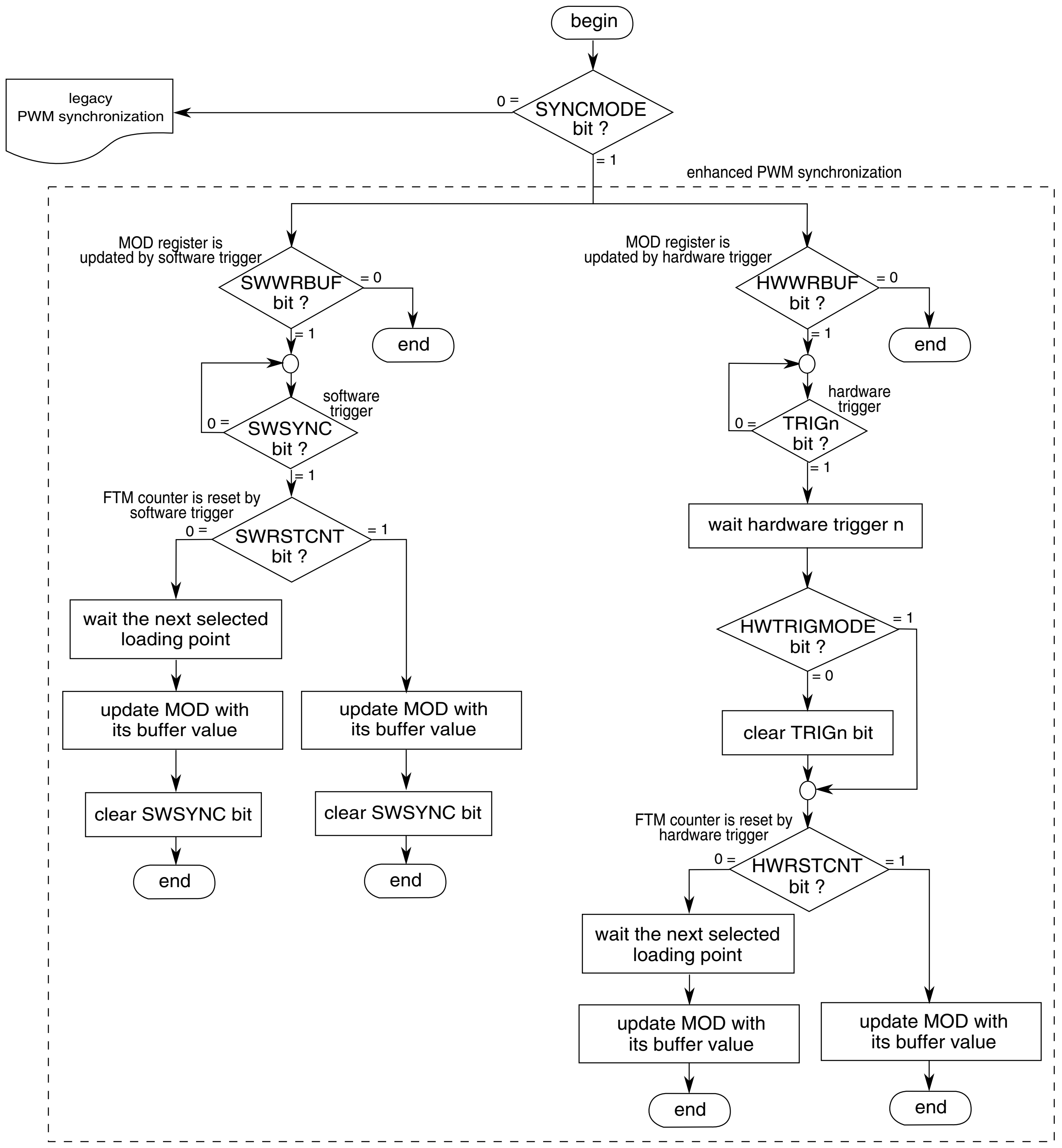

begin
SYNCMODE bit ?
0 =
legacy PWM synchronization
= 1
enhanced PWM synchronization
MOD register is updated by software trigger
SWWRBUF bit ?
= 0
end
= 1
software trigger
SWSYNC bit ?
0 =
= 1
FTM counter is reset by software trigger
SWRSTCNT bit ?
0 =
= 1
wait the next selected loading point
update MOD with its buffer value
update MOD with its buffer value
clear SWSYNC bit
clear SWSYNC bit
end
end
MOD register is updated by hardware trigger
HWWRBUF bit ?
= 0
end
= 1
hardware trigger
TRIGn bit ?
0 =
= 1
wait hardware trigger n
HWTRIGMODE bit ?
= 1
= 0
clear TRIGn bit
FTM counter is reset by hardware trigger
HWRSTCNT bit ?
0 =
= 1
wait the next selected loading point
update MOD with its buffer value
update MOD with its buffer value
end
end

Figure 26-164. MOD register synchronization flowchart

In the case of legacy PWM synchronization, the MOD register synchronization depends on PWMSYNC and REINIT bits according to the following description.

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 0), then this synchronization is made on the next selected loading point after an enabled trigger event takes place. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected loading point. If the trigger event was a hardware trigger, then the trigger enable bit (TRIGn) is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

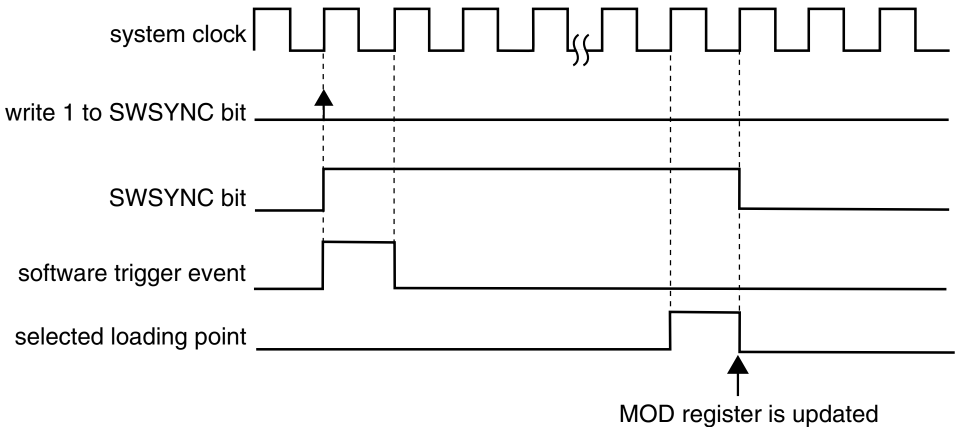


Figure 26-165. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 0), and software trigger was used

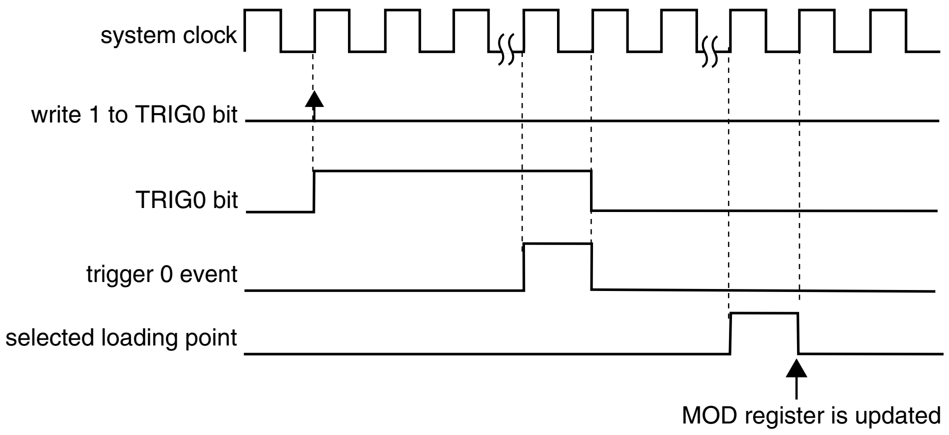


Figure 26-166. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 0), and a hardware trigger was used

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 1), then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

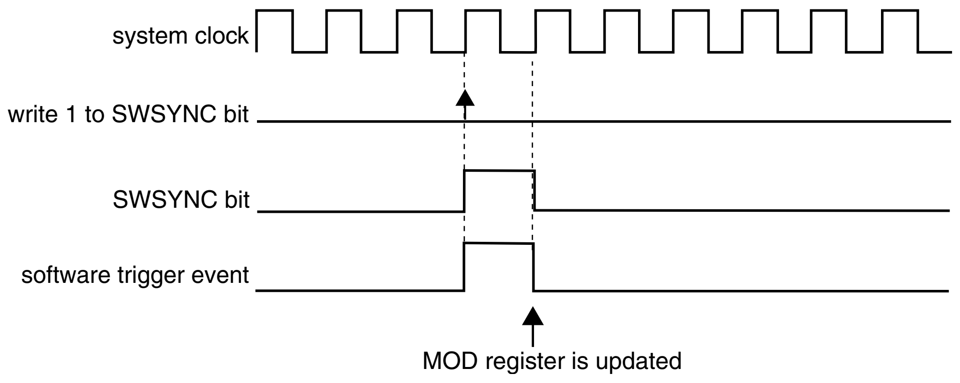


Figure 26-167. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 1), and software trigger was used

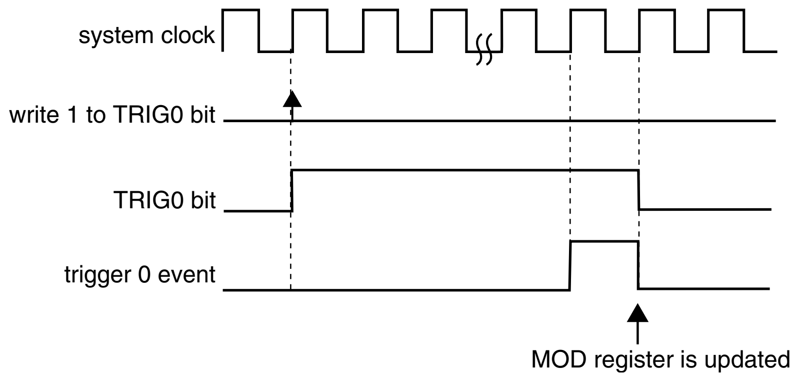


Figure 26-168. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 1), and a hardware trigger was used

If (SYNCMODE = 0) and (PWMSYNC = 1), then this synchronization is made on the next selected loading point after the software trigger event takes place. The SWSYNC bit is cleared on the next selected loading point:

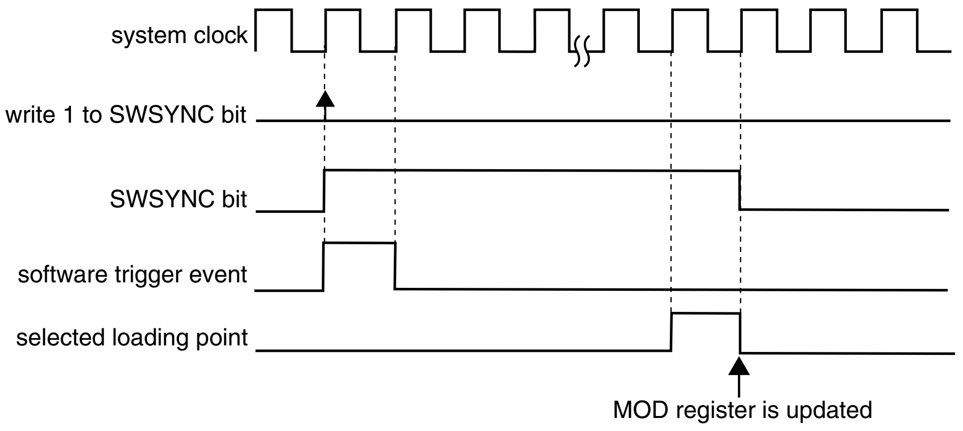


Figure 26-169. MOD synchronization with (SYNCMODE = 0) and (PWMSYNC = 1)

26.4.11.5 CNTIN register synchronization

The CNTIN register synchronization updates the CNTIN register with its buffer value.

This synchronization is enabled if (FTMEN = 1), (SYNCMODE = 1), and (CNTINC = 1). The CNTIN register synchronization can be done only by the enhanced PWM synchronization (SYNCMODE = 1). The synchronization mechanism is the same as the MOD register synchronization done by the enhanced PWM synchronization; see MOD register synchronization.

26.4.11.6 C(n)V and C(n+1)V register synchronization

The C(n)V and C(n+1)V registers synchronization updates the C(n)V and C(n+1)V registers with their buffer values.

This synchronization is enabled if (FTMEN = 1) and (SYNCEN = 1). The synchronization mechanism is the same as the MOD register synchronization. However, it is expected that the C(n)V and C(n+1)V registers be synchronized only by the enhanced PWM synchronization (SYNCMODE = 1).

26.4.11.7 OUTMASK register synchronization

The OUTMASK register synchronization updates the OUTMASK register with its buffer value.

The OUTMASK register can be updated at each rising edge of system clock (SYNCHOM = 0), by the enhanced PWM synchronization (SYNCHOM = 1 and SYNCMODE = 1) or by the legacy PWM synchronization (SYNCHOM = 1 and SYNCMODE = 0). However, it is expected that the OUTMASK register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the OUTMASK register synchronization depends on SWOM and HWOM bits. See the following flowchart:

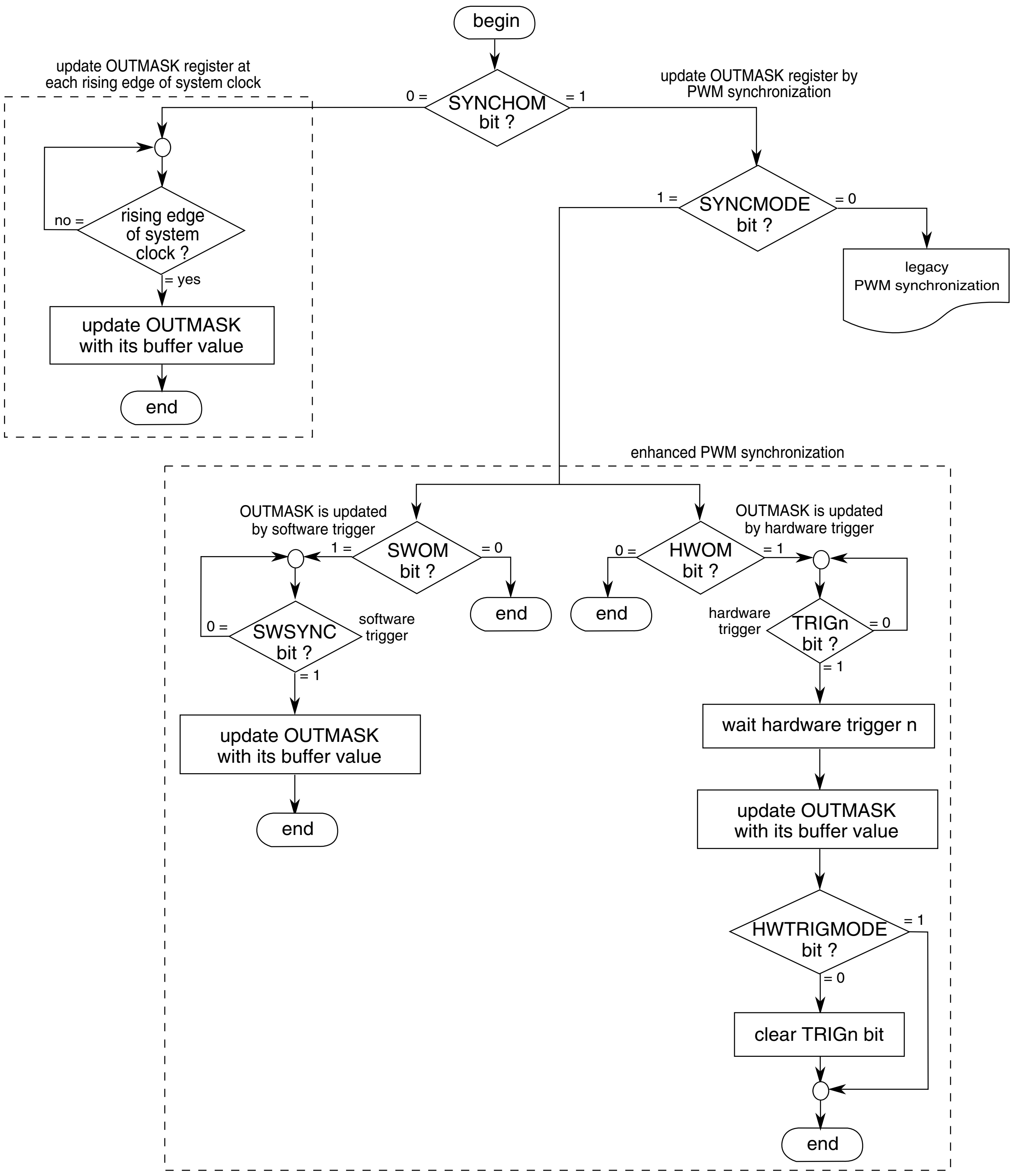

begin
SYNCHOM bit ?
0 =
= 1
update OUTMASK register at each rising edge of system clock
update OUTMASK register by PWM synchronization
rising edge of system clock ?
no =
= yes
update OUTMASK with its buffer value
end
SYNCMODE bit ?
1 =
= 0
legacy PWM synchronization
enhanced PWM synchronization
OUTMASK is updated by software trigger
SWOM bit ?
1 =
= 0
end
SWSYNC bit ?
0 =
= 1
software trigger
update OUTMASK with its buffer value
end
OUTMASK is updated by hardware trigger
HWOM bit ?
0 =
= 1
end
TRIGn bit ?
hardware trigger
= 0
= 1
wait hardware trigger n
update OUTMASK with its buffer value
HWTRIGMODE bit ?
= 1
= 0
clear TRIGn bit
end

Figure 26-170. OUTMASK register synchronization flowchart

In the case of legacy PWM synchronization, the OUTMASK register synchronization depends on PWMSYNC bit according to the following description.

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 0), then this synchronization is done on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected loading point. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

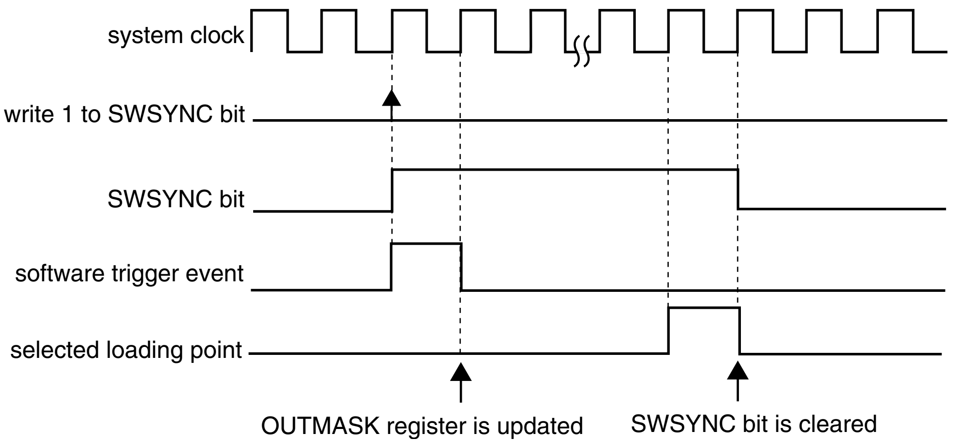


Figure 26-171. OUTMASK synchronization with (SYNCMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0) and software trigger was used

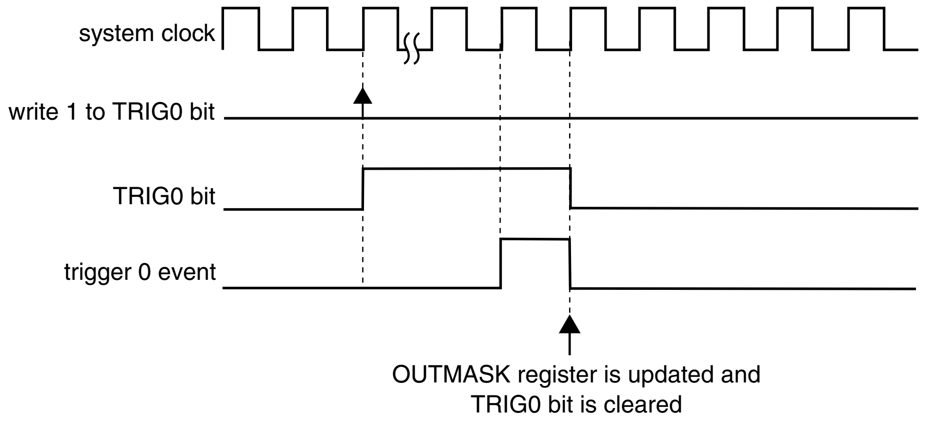


Figure 26-172. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0), and a hardware trigger was used

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 1), then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger. An example with a hardware trigger follows.

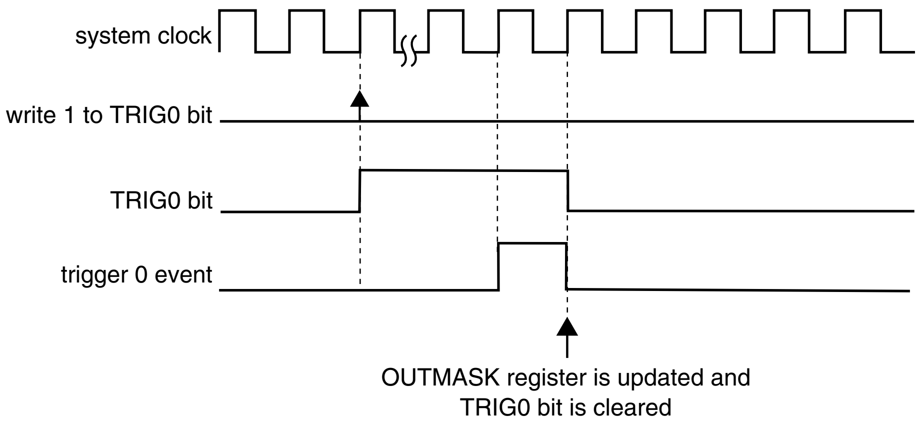


Figure 26-173. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 1), and a hardware trigger was used

26.4.11.8 INVCTRL register synchronization

The INVCTRL register synchronization updates the INVCTRL register with its buffer value.

The INVCTRL register can be updated at each rising edge of system clock (INVC = 0) or by the enhanced PWM synchronization (INVC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the INVCTRL register synchronization depends on SWINVC and HWINVC bits.

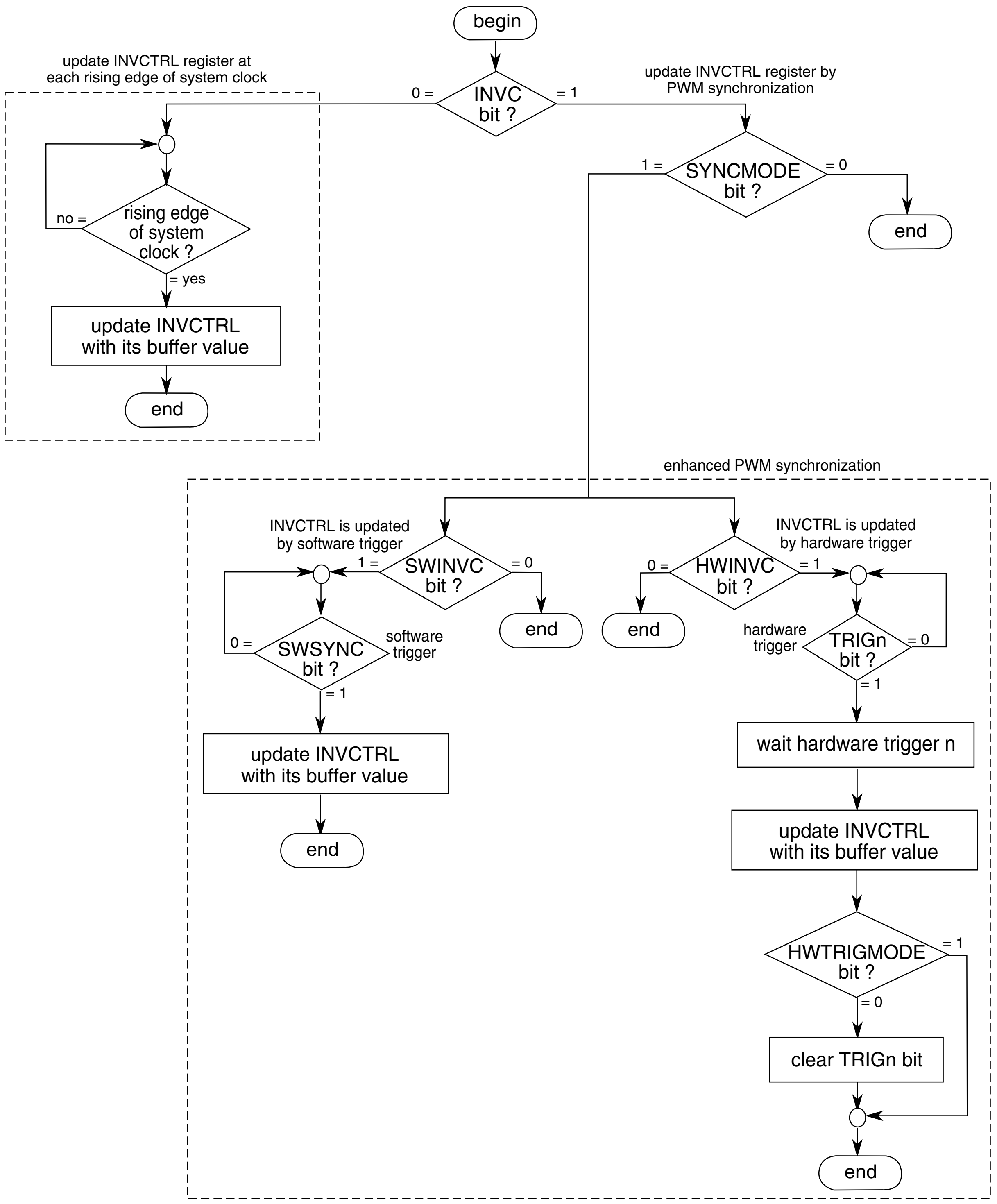

begin
INVC bit ?
0 =
= 1
update INVCTRL register at each rising edge of system clock
update INVCTRL register by PWM synchronization
rising edge of system clock ?
no =
= yes
update INVCTRL with its buffer value
end
SYNCMODE bit ?
1 =
= 0
end
enhanced PWM synchronization
INVCTRL is updated by software trigger
SWINVC bit ?
1 =
= 0
end
SWSYNC bit ?
0 =
= 1
software trigger
update INVCTRL with its buffer value
end
INVCTRL is updated by hardware trigger
HWINVC bit ?
0 =
= 1
end
hardware trigger
TRIGn bit ?
= 0
= 1
wait hardware trigger n
update INVCTRL with its buffer value
HWTRIGMODE bit ?
= 1
= 0
clear TRIGn bit
end

Figure 26-174. INVCTRL register synchronization flowchart

26.4.11.9 SWOCTRL register synchronization

The SWOCTRL register synchronization updates the SWOCTRL register with its buffer value.

The SWOCTRL register can be updated at each rising edge of system clock (SWOC = 0) or by the enhanced PWM synchronization (SWOC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the SWOCTRL register synchronization depends on SWSOC and HWSOC bits.

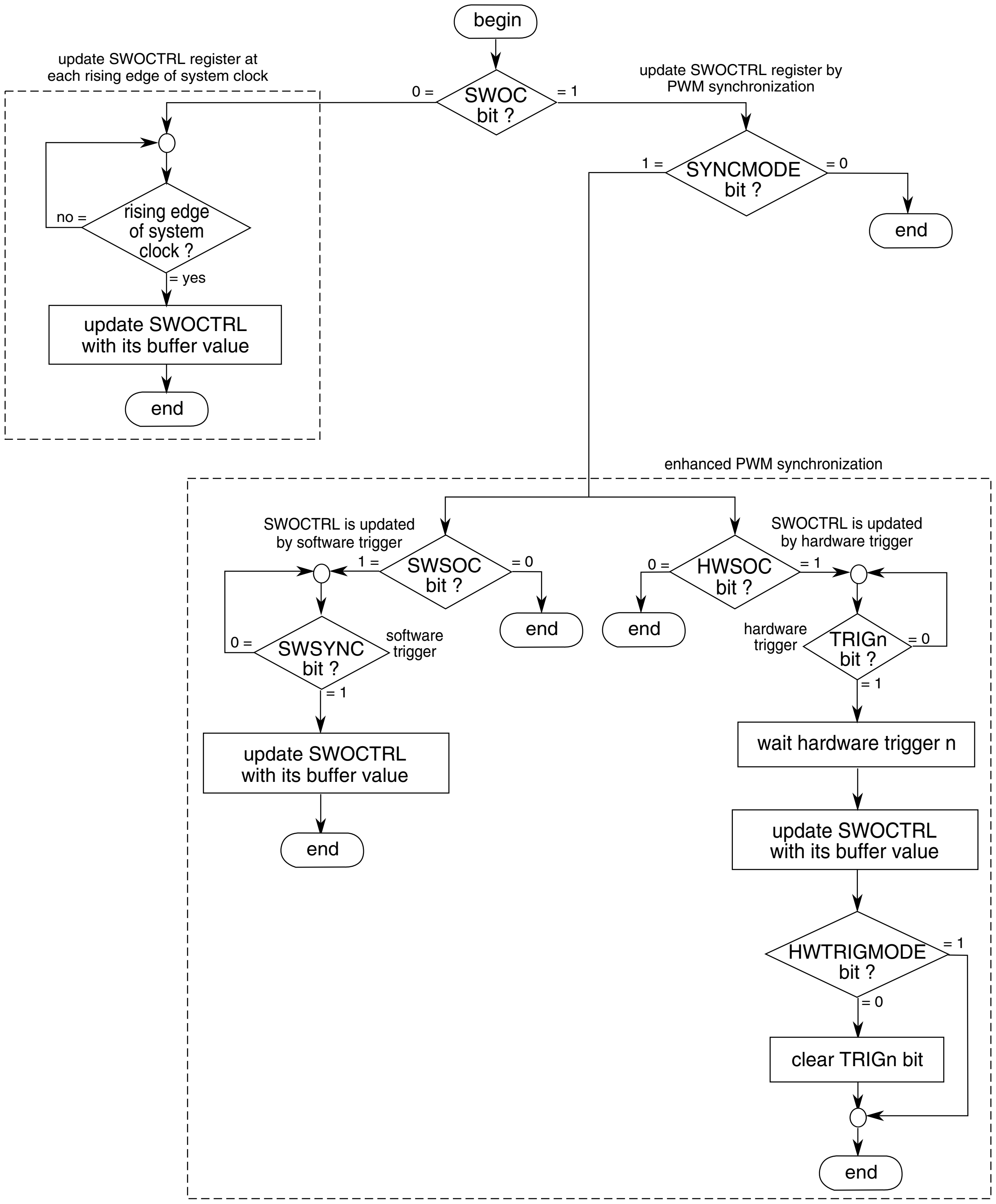

begin
SWOC bit ?
0 =
= 1
update SWOCTRL register at each rising edge of system clock
update SWOCTRL register by PWM synchronization
no =
rising edge of system clock ?
= yes
update SWOCTRL with its buffer value
end
1 =
SYNCMODE bit ?
= 0
end
enhanced PWM synchronization
SWOCTRL is updated by software trigger
1 =
SWSOC bit ?
= 0
end
0 =
SWSYNC bit ?
software trigger
= 1
update SWOCTRL with its buffer value
end
SWOCTRL is updated by hardware trigger
0 =
HWSOC bit ?
= 1
end
hardware trigger
TRIGn bit ?
= 0
= 1
wait hardware trigger n
update SWOCTRL with its buffer value
HWTRIGMODE bit ?
= 1
= 0
clear TRIGn bit
end

Figure 26-175. SWOCTRL register synchronization flowchart

26.4.11.10 FTM counter synchronization

The FTM counter synchronization is a mechanism that allows the FTM to restart the PWM generation at a certain point in the PWM period. The channels outputs are forced to their initial value, except for channels in Output Compare mode, and the FTM counter is forced to its initial counting value defined by CNTIN register.

The following figure shows the FTM counter synchronization. Note that after the synchronization event occurs, the channel (n) is set to its initial value and the channel (n +1) is not set to its initial value due to a specific timing of this figure in which the deadtime insertion prevents this channel output from transitioning to 1. If no deadtime insertion is selected, then the channel (n+1) transitions to logical value 1 immediately after the synchronization event occurs.

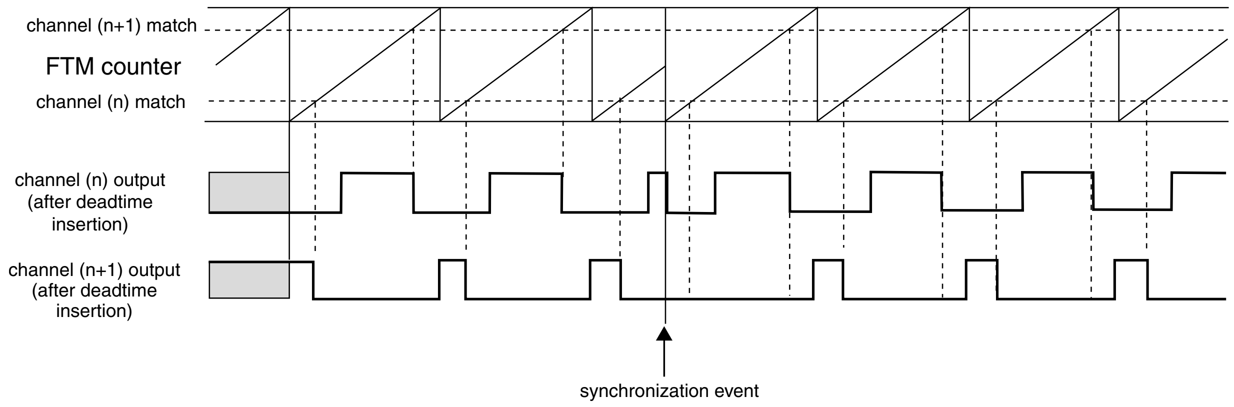

channel (n+1) match
FTM counter
channel (n) match
channel (n) output (after deadtime insertion)
channel (n+1) output (after deadtime insertion)
synchronization event

Figure 26-176. FTM counter synchronization

The FTM counter synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, the FTM counter must be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the FTM counter synchronization depends on SWRSTCNT and HWRSTCNT bits according to the following flowchart.

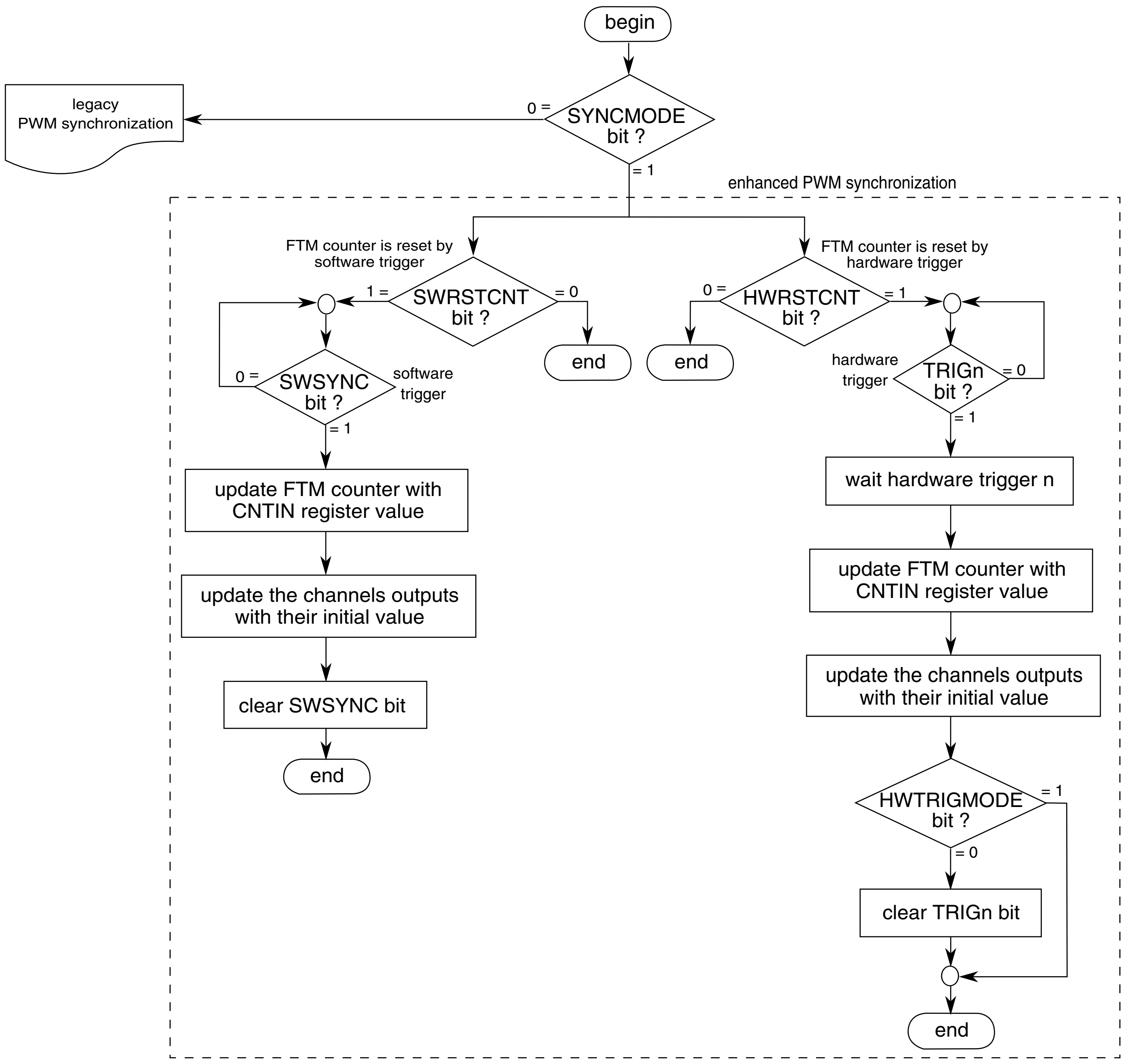

begin
SYNCMODE bit ?
0 =
legacy PWM synchronization
= 1
enhanced PWM synchronization
FTM counter is reset by software trigger
SWRSTCNT bit ?
1 =
= 0
end
SWSYNC bit ?
0 =
software trigger
= 1
update FTM counter with CNTIN register value
update the channels outputs with their initial value
clear SWSYNC bit
end
FTM counter is reset by hardware trigger
HWRSTCNT bit ?
0 =
= 1
end
hardware trigger
TRIGn bit ?
= 0
= 1
wait hardware trigger n
update FTM counter with CNTIN register value
update the channels outputs with their initial value
HWTRIGMODE bit ?
= 1
= 0
clear TRIGn bit
end

Figure 26-177. FTM counter synchronization flowchart

In the case of legacy PWM synchronization, the FTM counter synchronization depends on REINIT and PWMSYNC bits according to the following description.

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 0) then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

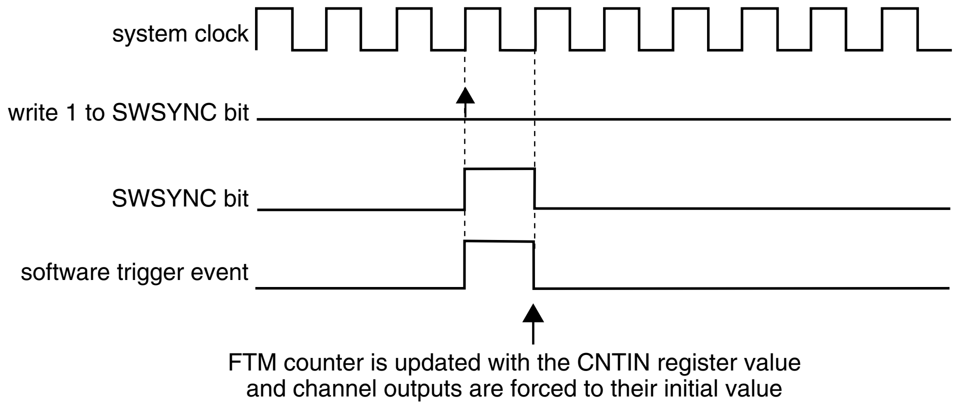

system clock
write 1 to SWSYNC bit
SWSYNC bit
software trigger event
FTM counter is updated with the CNTIN register value
and channel outputs are forced to their initial value

Figure 26-178. FTM counter synchronization with (SYNCMODE = 0), (REINIT = 1), (PWMSYNC = 0), and software trigger was used

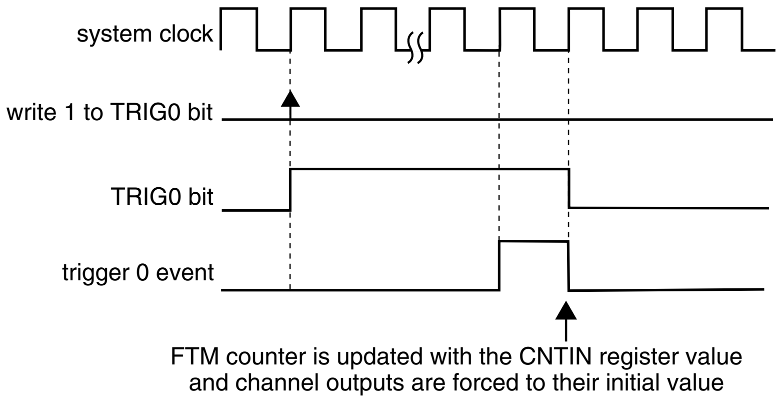

system clock
write 1 to TRIG0 bit
TRIG0 bit
trigger 0 event
FTM counter is updated with the CNTIN register value
and channel outputs are forced to their initial value

Figure 26-179. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 0), and a hardware trigger was used

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 1) then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger.

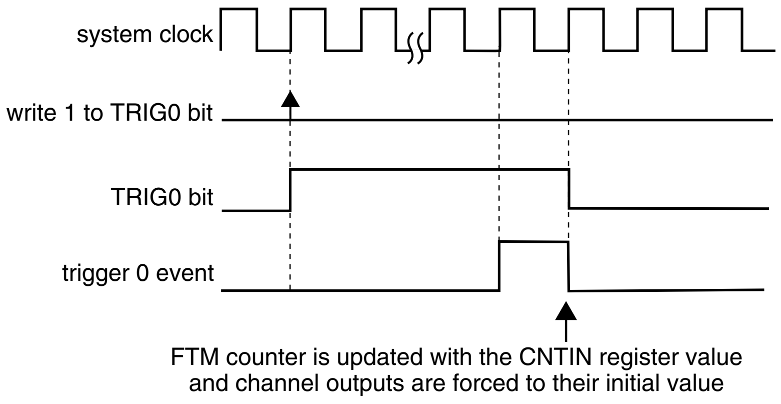

system clock
write 1 to TRIG0 bit
TRIG0 bit
trigger 0 event
FTM counter is updated with the CNTIN register value
and channel outputs are forced to their initial value

Figure 26-180. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 1), and a hardware trigger was used

26.4.12 Inverting

The invert functionality swaps the signals between channel (n) and channel (n+1) outputs. The inverting operation is selected when:

- FTMEN = 1
- DECAPEN = 0
- COMBINE = 1
- COMP = 1
- CPWMS = 0, and
- INVm = 1 (where m represents a channel pair)

The INVm bit in INVCTRL register is updated with its buffer value according to INVCTRL register synchronization

In High-True (ELSnB:ELSnA = 1:0) Combine mode, the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN), forced high at the channel (n) match and forced low at the channel (n+1) match. If the inverting is selected, the channel (n) output behavior is changed to force high at the beginning of the PWM period, force low at the channel (n) match and force high at the channel (n+1) match. See the following figure.

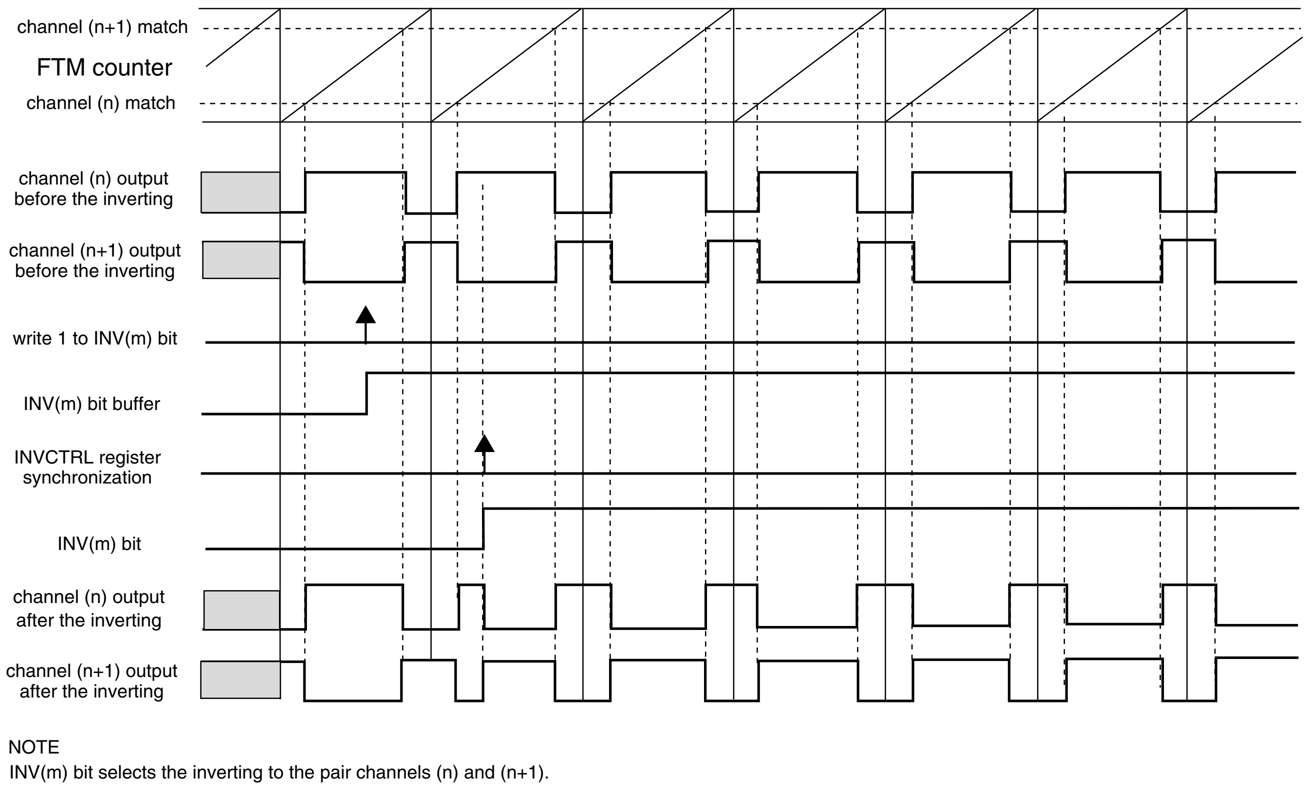

channel (n+1) match
FTM counter
channel (n) match
channel (n) output before the inverting
channel (n+1) output before the inverting
write 1 to INV(m) bit
INV(m) bit buffer
INVCTRL register synchronization
INV(m) bit
channel (n) output after the inverting
channel (n+1) output after the inverting
NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

Figure 26-181. Channels (n) and (n+1) outputs after the inverting in High-True (ELSnB:ELSnA = 1:0) Combine mode

Note that the ELSnB:ELSnA bits value should be considered because they define the active state of the channels outputs. In Low-True (ELSnB:ELSnA = X:1) Combine mode, the channel (n) output is forced high at the beginning of the period, forced low at the channel (n) match and forced high at the channel (n+1) match. When inverting is selected, the channels (n) and (n+1) present waveforms as shown in the following figure.

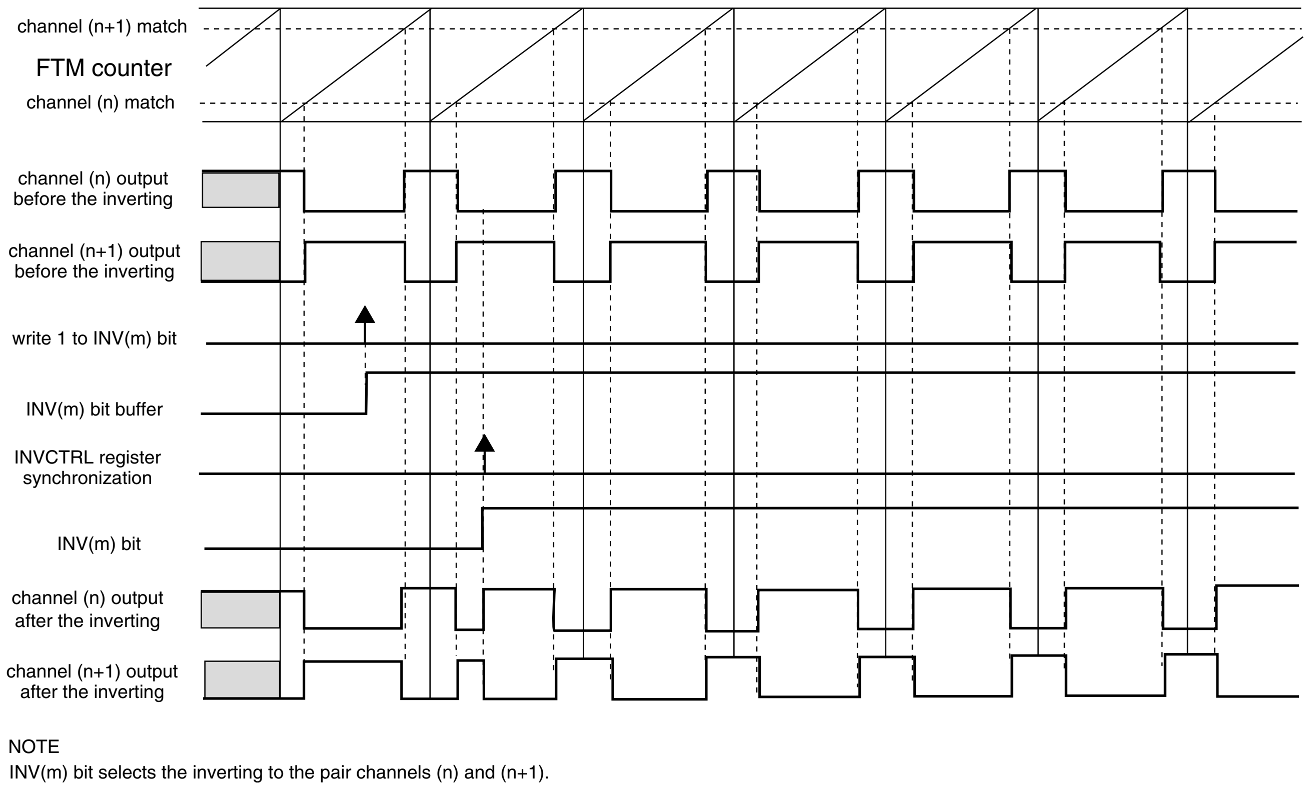

channel (n+1) match
FTM counter
channel (n) match
channel (n) output before the inverting
channel (n+1) output before the inverting
write 1 to INV(m) bit
INV(m) bit buffer
INVCTRL register synchronization
INV(m) bit
channel (n) output after the inverting
channel (n+1) output after the inverting
NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

Figure 26-182. Channels (n) and (n+1) outputs after the inverting in Low-True (ELSnB:ELSnA = X:1) Combine mode

Note

The inverting feature must be used only in Combine mode.

26.4.13 Software output control

The software output control forces the channel output according to software defined values at a specific time in the PWM generation.

The software output control is selected when:

- FTMEN = 1
- DECAPEN = 0
- COMBINE = 1
- CPWMS = 0, and
- CHnOC = 1

The CHnOC bit enables the software output control for a specific channel output and the CHnOCV selects the value that is forced to this channel output.

Both CHnOC and CHnOCV bits in SWOCTRL register are buffered and updated with their buffer value according to SWOCTRL register synchronization.

The following figure shows the channels (n) and (n+1) outputs signals when the software output control is used. In this case the channels (n) and (n+1) are set to Combine and Complementary mode.

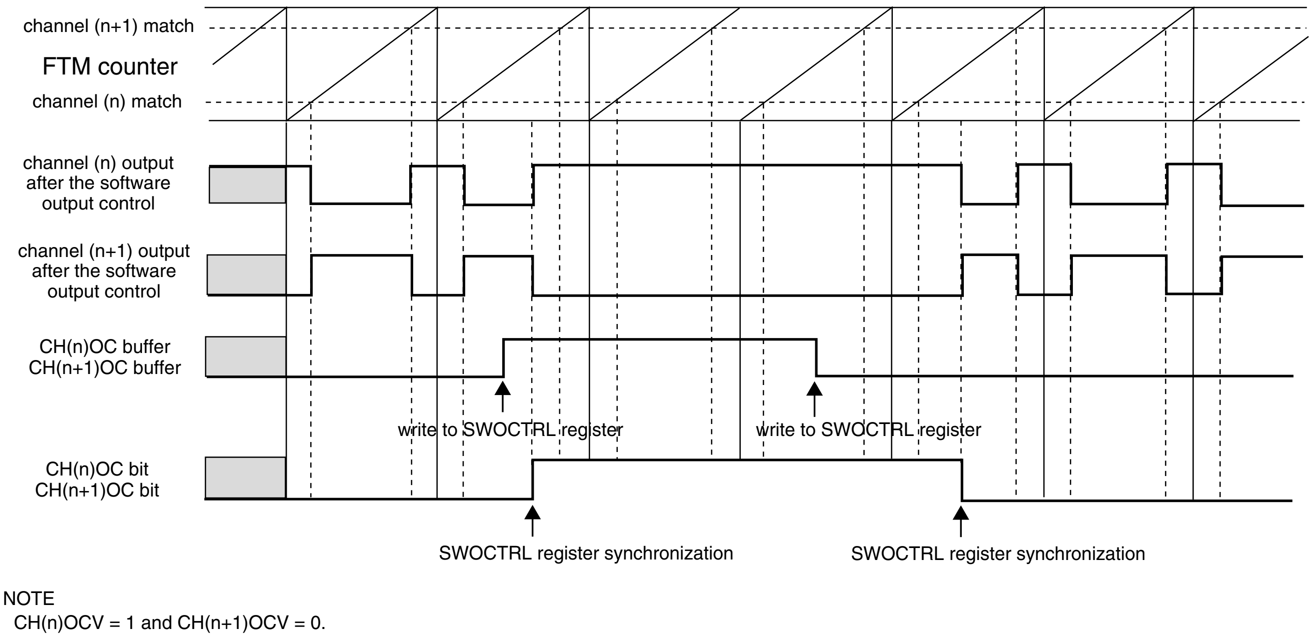

channel (n+1) match
FTM counter
channel (n) match
channel (n) output after the software output control
channel (n+1) output after the software output control
CH(n)OC buffer CH(n+1)OC buffer
write to SWOCTRL register
write to SWOCTRL register
CH(n)OC bit CH(n+1)OC bit
SWOCTRL register synchronization
SWOCTRL register synchronization
NOTE
CH(n)OCV = 1 and CH(n+1)OCV = 0.

Figure 26-183. Example of software output control in Combine and Complementary mode

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is zero.

Table 26-183. Software ouput control behavior when (COMP = 0)

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	Channel (n) Output	Channel (n+1) Output
0	0	X	X	is not modified by SWOC	is not modified by SWOC
1	1	0	0	is forced to zero	is forced to zero
1	1	0	1	is forced to zero	is forced to one
1	1	1	0	is forced to one	is forced to zero
1	1	1	1	is forced to one	is forced to one

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is one.

Table 26-184. Software ouput control behavior when (COMP = 1)

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	Channel (n) Output	Channel (n+1) Output
0	0	X	X	is not modified by SWOC	is not modified by SWOC
1	1	0	0	is forced to zero	is forced to zero
1	1	0	1	is forced to zero	is forced to one

Table continues on the next page...

Table 26-184. Software ouput control behavior when (COMP = 1) (continued)

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	Channel (n) Output	Channel (n+1) Output
1	1	1	0	is forced to one	is forced to zero
1	1	1	1	is forced to one	is forced to zero

Note

- The software output control feature must be used only in Combine mode.
- The CH(n)OC and CH(n+1)OC bits should be equal.
- The COMP bit must not be modified when software output control is enabled, that is, CH(n)OC = 1 and/or CH(n +1)OC = 1.
- Software output control has the same behavior with disabled or enabled FTM counter (see the CLKS field description in the Status and Control register).

26.4.14 Deadtime insertion

The deadtime insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non- zero).

DEADTIME register defines the deadtime delay that can be used for all FTM channels. The DTPS[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the deadtime modulo, that is, the number of the deadtime prescaler clocks.

The deadtime delay insertion ensures that no two complementary signals (channels (n) and (n+1)) drive the active state at the same time.

If POL(n) = 0, POL(n+1) = 0, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the low value until the end of the deadtime delay when the channel (n) output is set. Similarly, when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the low value until the end of the deadtime delay when the channel (n+1) output is set. See the following figures.

If POL(n) = 1, POL(n+1) = 1, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the high value until the end of the deadtime delay when the channel (n) output is cleared. Similarly, when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the high value until the end of the deadtime delay when the channel (n +1) output is cleared.

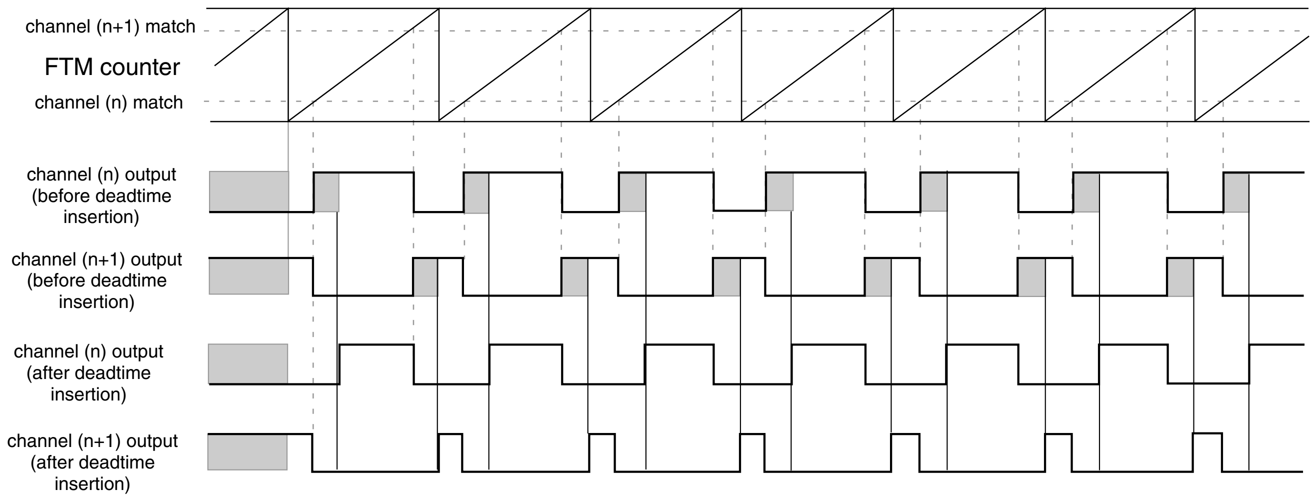

channel (n+1) match
FTM counter
channel (n) match
channel (n) output (before deadtime insertion)
channel (n+1) output (before deadtime insertion)
channel (n) output (after deadtime insertion)
channel (n+1) output (after deadtime insertion)

Figure 26-184. Deadtime insertion with ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0

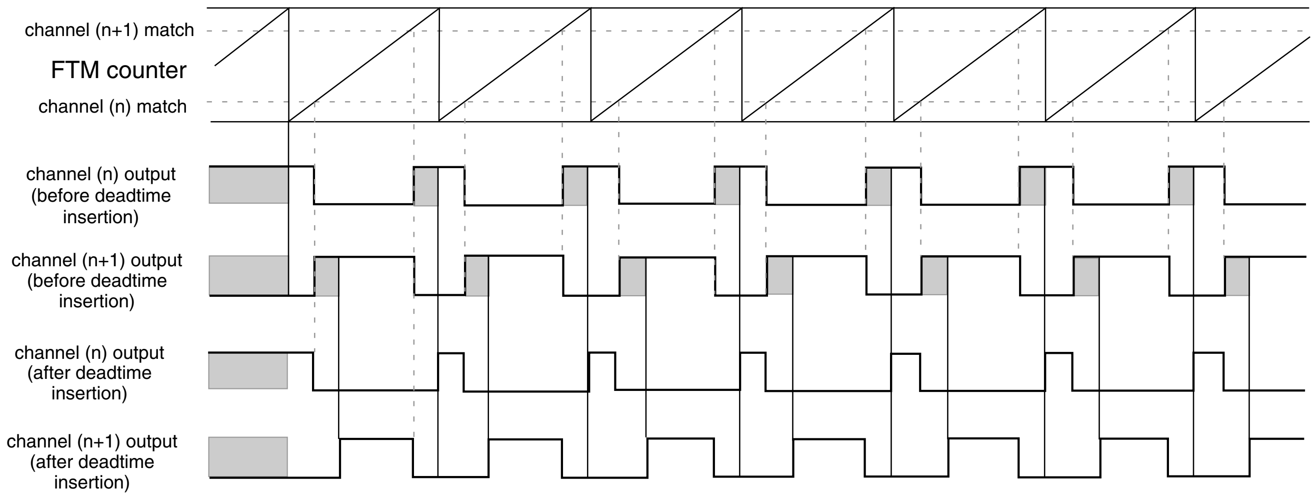

channel (n+1) match
FTM counter
channel (n) match
channel (n) output (before deadtime insertion)
channel (n+1) output (before deadtime insertion)
channel (n) output (after deadtime insertion)
channel (n+1) output (after deadtime insertion)

Figure 26-185. Deadtime insertion with ELSnB:ELSnA = X:1, POL(n) = 0, and POL(n+1) = 0

NOTE

The deadtime feature must be used only in Combine and Complementary modes.

26.4.14.1 Deadtime insertion corner cases

If (PS[2:0] is cleared), (DTPS[1:0] = 0:0 or DTPS[1:0] = 0:1):

- and the deadtime delay is greater than or equal to the channel (n) duty cycle ((C(n +1)V – C(n)V) × system clock), then the channel (n) output is always the inactive value (POL(n) bit value).
- and the deadtime delay is greater than or equal to the channel (n+1) duty cycle ((MOD – CNTIN + 1 – (C(n+1)V – C(n)V)) × system clock), then the channel (n+1) output is always the inactive value (POL(n+1) bit value).

Although, in most cases the deadtime delay is not comparable to channels (n) and (n+1) duty cycle, the following figures show examples where the deadtime delay is comparable to the duty cycle.

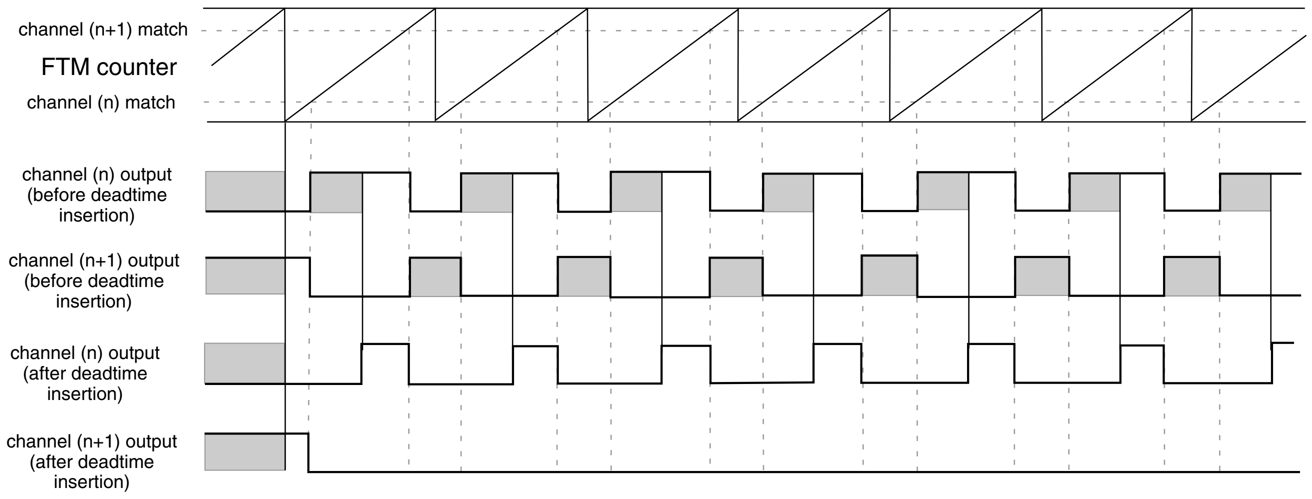

channel (n+1) match
FTM counter
channel (n) match
channel (n) output (before deadtime insertion)
channel (n+1) output (before deadtime insertion)
channel (n) output (after deadtime insertion)
channel (n+1) output (after deadtime insertion)

Figure 26-186. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channel (n+1) duty cycle

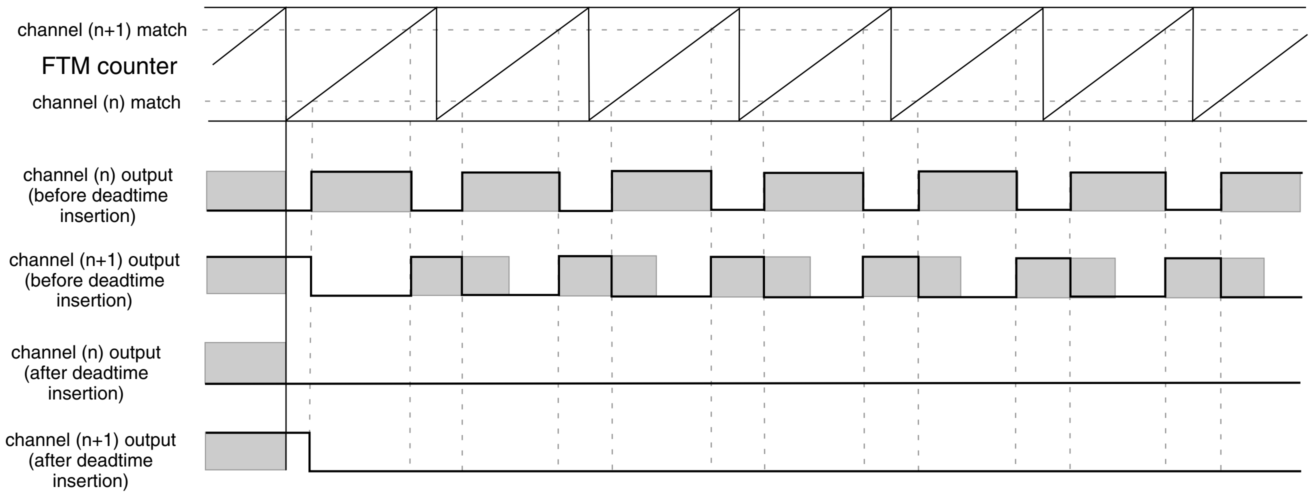

channel (n+1) match
FTM counter
channel (n) match
channel (n) output (before deadtime insertion)
channel (n+1) output (before deadtime insertion)
channel (n) output (after deadtime insertion)
channel (n+1) output (after deadtime insertion)

Figure 26-187. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channels (n) and (n+1) duty cycle

26.4.15 Output mask

The output mask can be used to force channels output to their inactive state through software. For example: to control a BLDC motor.

Any write to the OUTMASK register updates its write buffer. The OUTMASK register is updated with its buffer value by PWM synchronization; see OUTMASK register synchronization.

If CHnOM = 1, then the channel (n) output is forced to its inactive state (POLn bit value). If CHnOM = 0, then the channel (n) output is unaffected by the output mask. See the following figure.

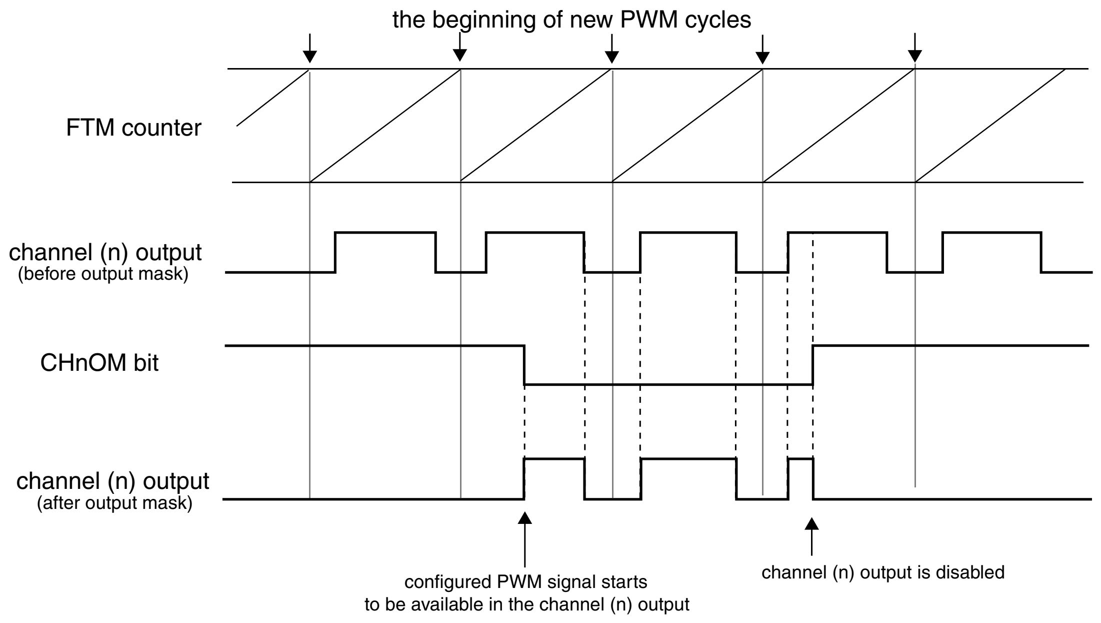

the beginning of new PWM cycles
FTM counter
channel (n) output
(before output mask)
CHnOM bit
channel (n) output
(after output mask)
configured PWM signal starts
to be available in the channel (n) output
channel (n) output is disabled

Figure 26-188. Output mask with POLn = 0

The following table shows the output mask result before the polarity control.

Table 26-185. Output mask result for channel (n) before the polarity control

CHnOM	Output Mask Input	Output Mask Result
0	inactive state	inactive state
	active state	active state
1	inactive state	inactive state
	active state	

Note

The output mask feature must be used only in Combine mode.

26.4.16 Fault control

The fault control is enabled if (FTMEN = 1) and (FAULTM[1:0] ≠ 0:0).

FTM can have up to four fault inputs. FAULTnEN bit (where n = 0, 1, 2, 3) enables the fault input n and FFLTRnEN bit enables the fault input n filter. FFVAL[3:0] bits select the value of the enabled filter in each enabled fault input.

First, each fault input signal is synchronized by the system clock; see the synchronizer block in the following figure. Following synchronization, the fault input n signal enters the filter block. When there is a state change in the fault input n signal, the 5-bit counter is reset and starts counting up. As long as the new state is stable on the fault input n, the counter continues to increment. If the 5-bit counter overflows, that is, the counter exceeds the value of the FFVAL[3:0] bits, the new fault input n value is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the fault input n signal before validation (counter overflow), the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by FFVAL[3:0] bits (× system clock) is regarded as a glitch and is not passed on to the edge detector.

The fault input n filter is disabled when the FFVAL[3:0] bits are zero or when FAULTnEN = 0. In this case, the fault input n signal is delayed 2 rising edges of the system clock and the FAULTFn bit is set on 3th rising edge of the system clock after a rising edge occurs on the fault input n.

If FFVAL[3:0] ≠ 0000 and FAULTnEN = 1, then the fault input n signal is delayed (3 + FFVAL[3:0]) rising edges of the system clock, that is, the FAULTFn bit is set (4 + FFVAL[3:0]) rising edges of the system clock after a rising edge occurs on the fault input n.

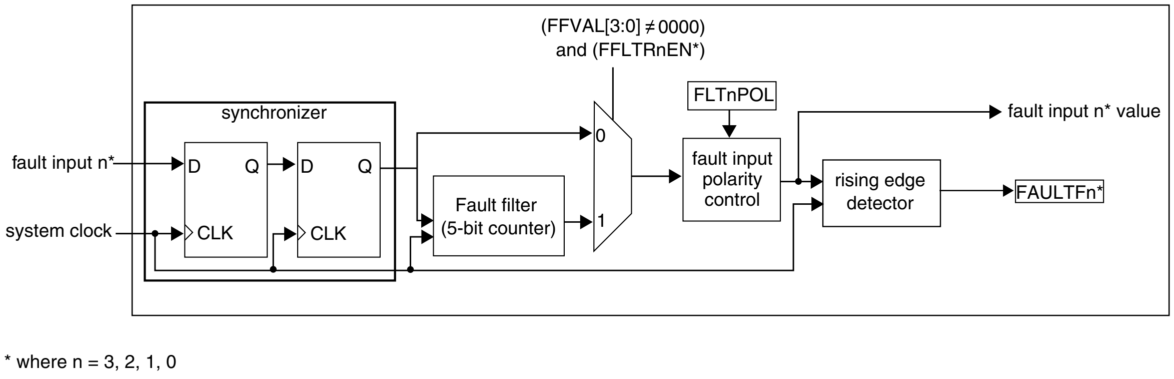


Figure 26-189. Fault input n control block diagram

If the fault control and fault input n are enabled and a rising edge at the fault input n signal is detected, a fault condition has occurred and the FAULTFn bit is set. The FAULTF bit is the logic OR of FAULTFn[3:0] bits. See the following figure.

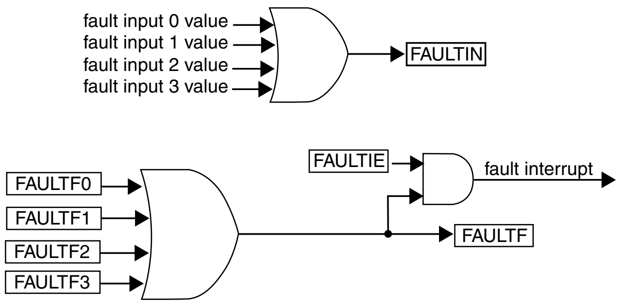


Figure 26-190. FAULTF and FAULTIN bits and fault interrupt

If the fault control is enabled (FAULTM[1:0] ≠ 0:0), a fault condition has occurred and (FAULTEN = 1), then outputs are forced to their safe values:

- Channel (n) output takes the value of POL(n)
- Channel (n+1) takes the value of POL(n+1)

The fault interrupt is generated when (FAULTF = 1) and (FAULTIE = 1). This interrupt request remains set until:

- Software clears the FAULTF bit by reading FAULTF bit as 1 and writing 0 to it
- Software clears the FAULTIE bit
- A reset occurs

Note

The fault control must be used only in Combine mode.

26.4.16.1 Automatic fault clearing

If the automatic fault clearing is selected (FAULTM[1:0] = 1:1), then the channels output disabled by fault control is again enabled when the fault input signal (FAULTIN) returns to zero and a new PWM cycle begins. See the following figure.

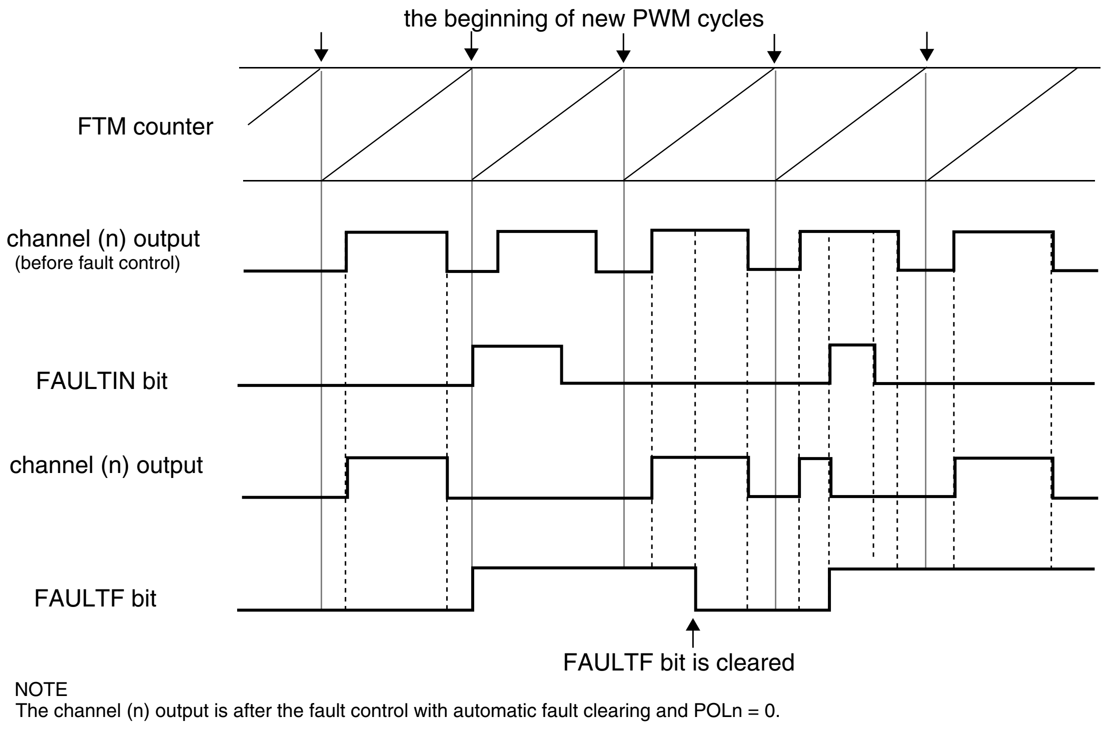
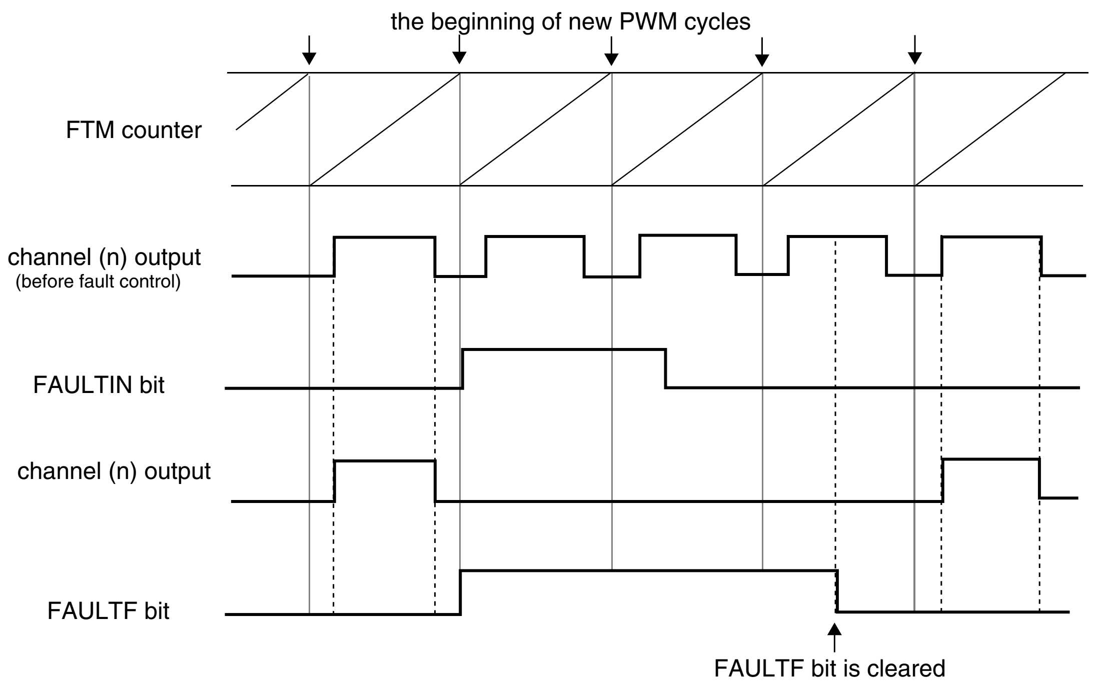

the beginning of new PWM cycles
FTM counter
channel (n) output
(before fault control)
FAULTIN bit
channel (n) output
FAULTF bit
FAULTF bit is cleared
NOTE
The channel (n) output is after the fault control with automatic fault clearing and POLn = 0.

Figure 26-191. Fault control with automatic fault clearing

26.4.16.2 Manual fault clearing

If the manual fault clearing is selected (FAULTM[1:0] = 0:1 or 1:0), then the channels output disabled by fault control is again enabled when the FAULTF bit is cleared and a new PWM cycle begins. See the following figure.



NOTE
The channel (n) output is after the fault control with manual fault clearing and POLn = 0.

Figure 26-192. Fault control with manual fault clearing

26.4.16.3 Fault inputs polarity control

The FLTjPOL bit selects the fault input j polarity, where j = 0, 1, 2, 3:

- If FLTjPOL = 0, the fault j input polarity is high, so the logical one at the fault input j indicates a fault.
- If FLTjPOL = 1, the fault j input polarity is low, so the logical zero at the fault input j indicates a fault.

26.4.17 Polarity control

The POLn bit selects the channel (n) output polarity:

- If POLn = 0, the channel (n) output polarity is high, so the logical one is the active state and the logical zero is the inactive state.
- If POLn = 1, the channel (n) output polarity is low, so the logical zero is the active state and the logical one is the inactive state.

Note

The polarity control must be used only in Combine mode.

26.4.18 Initialization

The initialization forces the CHnOI bit value to the channel (n) output when a one is written to the INIT bit.

The initialization depends on COMP and DTEN bits. The following table shows the values that channels (n) and (n+1) are forced by initialization when the COMP and DTEN bits are zero.

Table 26-186. Initialization behavior when (COMP = 0 and DTEN = 0)

CH(n)OI	CH(n+1)OI	Channel (n) Output	Channel (n+1) Output
0	0	is forced to zero	is forced to zero
0	1	is forced to zero	is forced to one
1	0	is forced to one	is forced to zero
1	1	is forced to one	is forced to one

The following table shows the values that channels (n) and (n+1) are forced by initialization when (COMP = 1) or (DTEN = 1).

Table 26-187. Initialization behavior when (COMP = 1 or DTEN = 1)

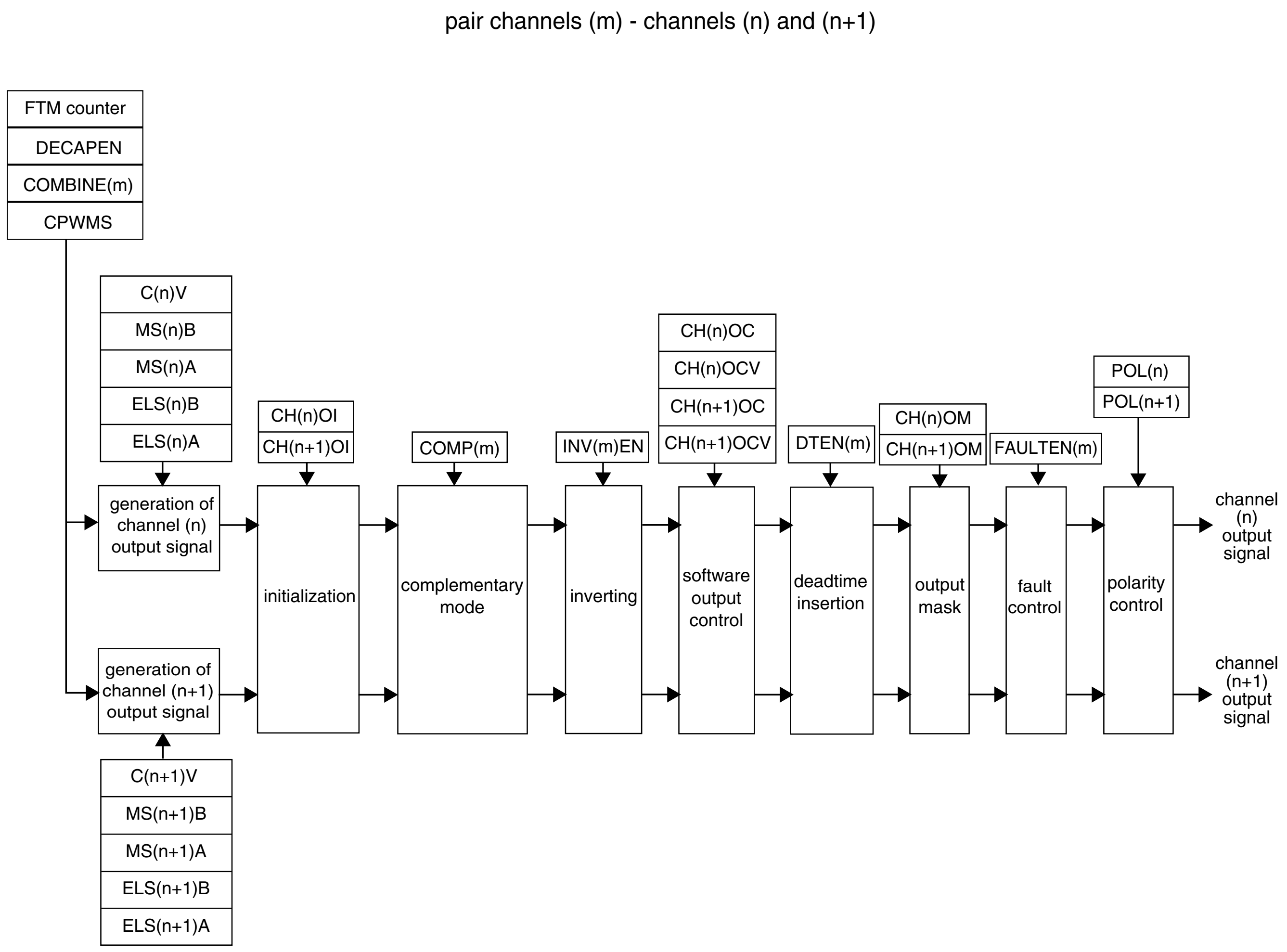
CH(n)OI	CH(n+1)OI	Channel (n) Output	Channel (n+1) Output
0	X	is forced to zero	is forced to one
1	X	is forced to one	is forced to zero

Note

The initialization feature must be used only in Combine mode and with disabled FTM counter. See the description of the CLKS field in the Status and Control register.

26.4.19 Features priority

The following figure shows the priority of the features used at the generation of channels (n) and (n+1) outputs signals.



NOTE
The channels (n) and (n+1) are in output compare, EPWM, CPWM or combine modes.

Figure 26-193. Priority of the features used at the generation of channels (n) and (n+1) outputs signals

Note

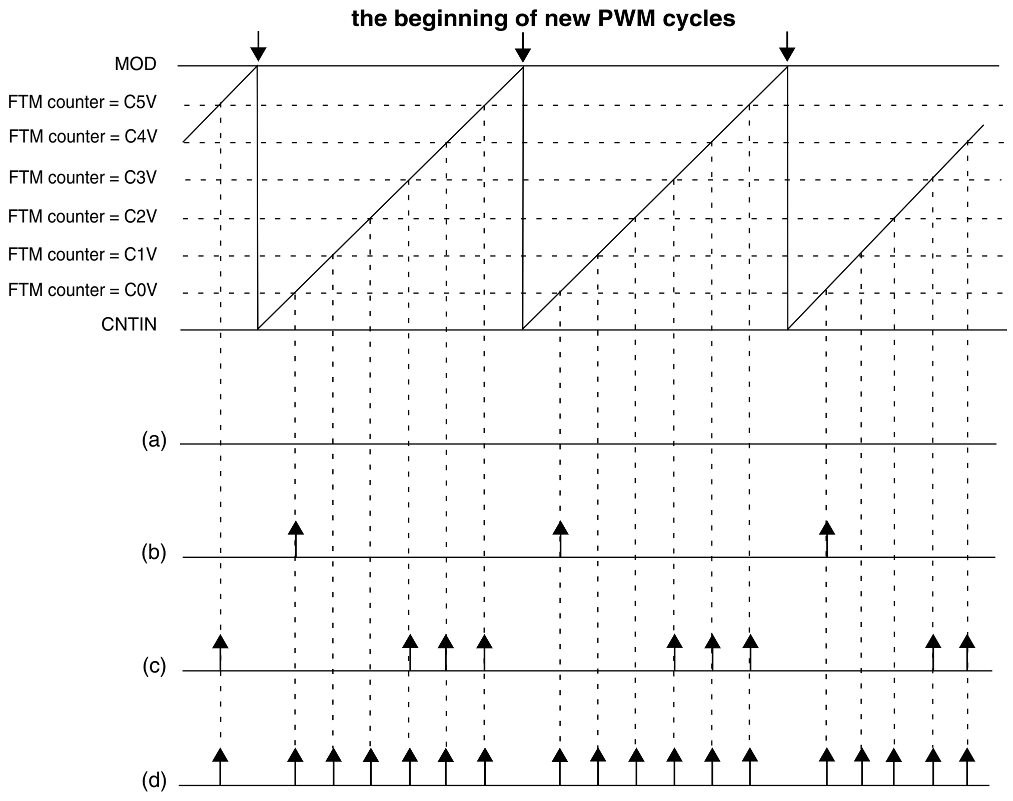
The Initialization feature must not be used with Inverting and Software output control features.

26.4.20 Channel trigger output

If CHjTRIG = 1, where j = 0, 1, 2, 3, 4, or 5, then the FTM generates a trigger when the channel (j) match occurs (FTM counter = C(j)V).

The channel trigger output provides a trigger signal that is used for on-chip modules.

The FTM is able to generate multiple triggers in one PWM period. Because each trigger is generated for a specific channel, several channels are required to implement this functionality. This behavior is described in the following figure.



NOTE

(a) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(b) CH0TRIG = 1, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(c) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1
(d) CH0TRIG = 1, CH1TRIG = 1, CH2TRIG = 1, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1

Figure 26-194. Channel match trigger

Note

The channel match trigger must be used only in Combine mode.

26.4.21 Initialization trigger

If INITTRIGEN = 1, then the FTM generates a trigger when the FTM counter is updated with the CNTIN register value in the following cases.

- The FTM counter is automatically updated with the CNTIN register value by the selected counting mode.
- When there is a write to CNT register.

- When there is the FTM counter synchronization.
- If (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits.

The following figures show these cases.

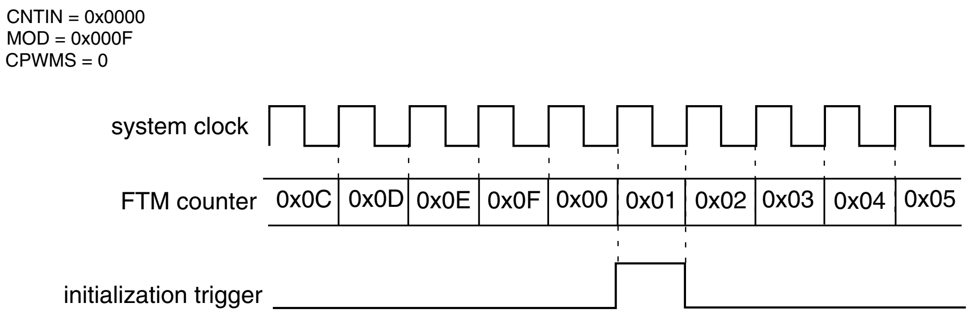


Figure 26-195. Initialization trigger is generated when the FTM counting achieves the CNTIN register value

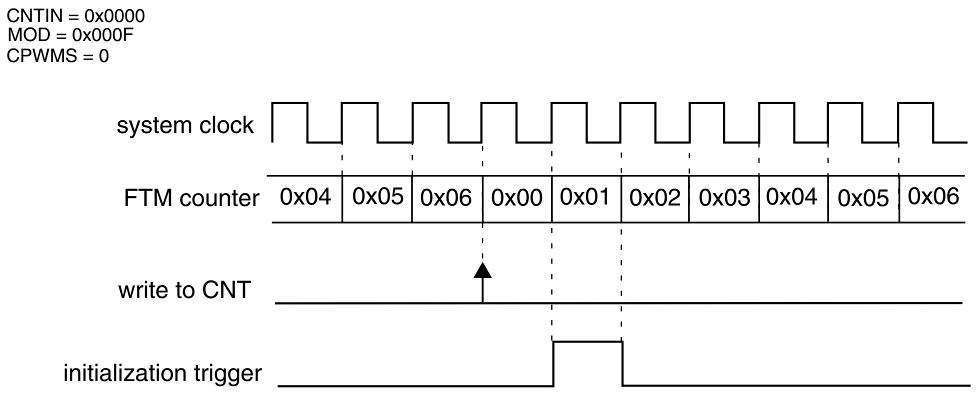


Figure 26-196. Initialization trigger is generated when there is a write to CNT register

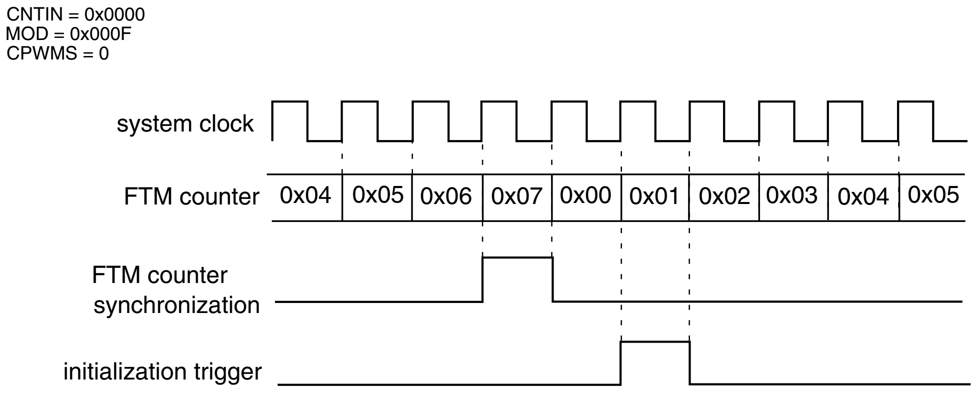


Figure 26-197. Initialization trigger is generated when there is the FTM counter synchronization

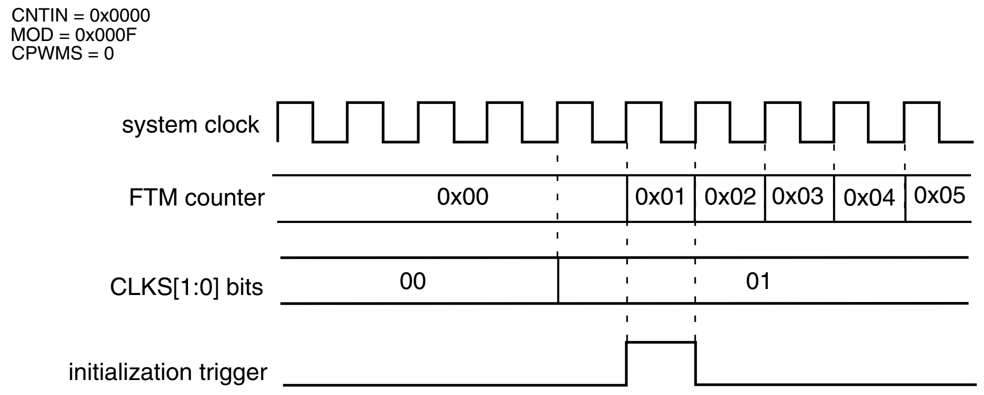


Figure 26-198. Initialization trigger is generated if (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits

The initialization trigger output provides a trigger signal that is used for on-chip modules.

Note

The initialization trigger must be used only in Combine mode.

26.4.22 Capture Test mode

The Capture Test mode allows to test the CnV registers, the FTM counter and the interconnection logic between the FTM counter and CnV registers.

In this test mode, all channels must be configured for Input Capture mode and FTM counter must be configured to the Up counting.

When the Capture Test mode is enabled (CAPTEST = 1), the FTM counter is frozen and any write to CNT register updates directly the FTM counter; see the following figure. After it was written, all CnV registers are updated with the written value to CNT register and CHnF bits are set. Therefore, the FTM counter is updated with its next value according to its configuration. Its next value depends on CNTIN, MOD, and the written value to FTM counter.

The next reads of CnV registers return the written value to the FTM counter and the next reads of CNT register return FTM counter next value.

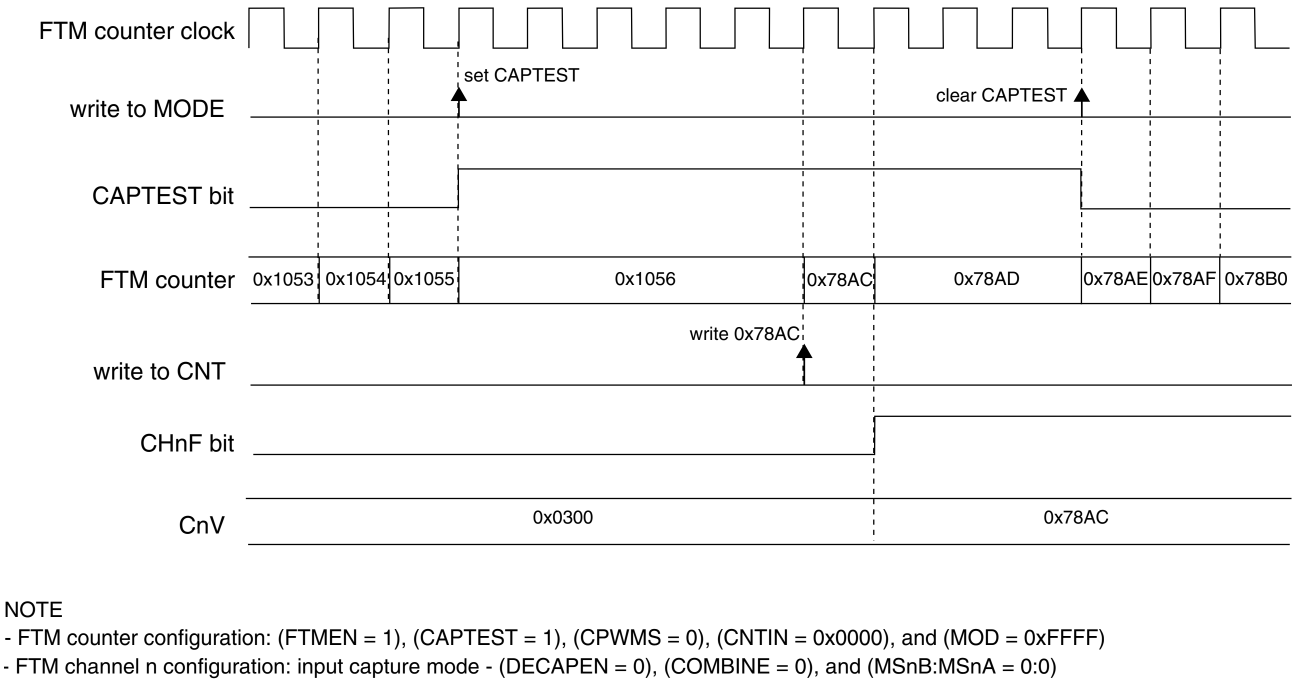

FTM counter clock
write to MODE
set CAPTEST
clear CAPTEST
CAPTEST bit
FTM counter
0x1053
0x1054
0x1055
0x1056
0x78AC
0x78AD
0x78AE
0x78AF
0x78B0
write 0x78AC
write to CNT
CHnF bit
CnV
0x0300
0x78AC
NOTE
- FTM counter configuration: (FTMEN = 1), (CAPTEST = 1), (CPWMS = 0), (CNTIN = 0x0000), and (MOD = 0xFFFF)
- FTM channel n configuration: input capture mode - (DECAPEN = 0), (COMBINE = 0), and (MSnB:MSnA = 0:0)

Figure 26-199. Capture Test mode

26.4.23 Dual Edge Capture mode

The Dual Edge Capture mode is selected if FTMEN = 1 and DECAPEN = 1. This mode allows to measure a pulse width or period of the signal on the input of channel (n) of a channel pair. The channel (n) filter can be active in this mode when n is 0 or 2.

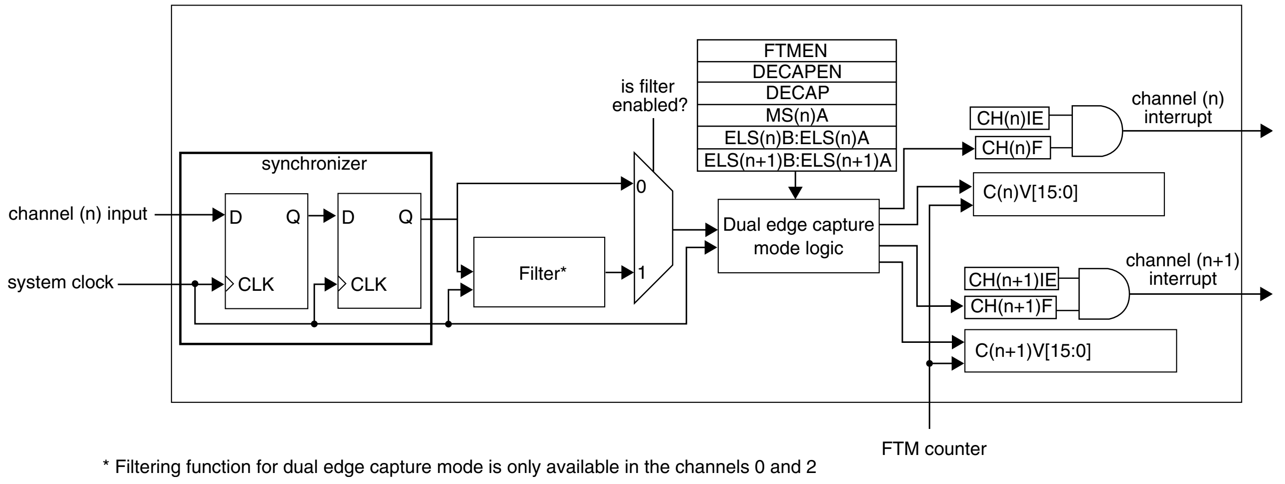


Figure 26-200. Dual Edge Capture mode block diagram

The MS(n)A bit defines if the Dual Edge Capture mode is one-shot or continuous.

The ELS(n)B:ELS(n)A bits select the edge that is captured by channel (n), and ELS(n +1)B:ELS(n+1)A bits select the edge that is captured by channel (n+1). If both ELS(n)B:ELS(n)A and ELS(n+1)B:ELS(n+1)A bits select the same edge, then it is the period measurement. If these bits select different edges, then it is a pulse width measurement.

In the Dual Edge Capture mode, only channel (n) input is used and channel (n+1) input is ignored.

If the selected edge by channel (n) bits is detected at channel (n) input, then CH(n)F bit is set and the channel (n) interrupt is generated (if CH(n)IE = 1). If the selected edge by channel (n+1) bits is detected at channel (n) input and (CH(n)F = 1), then CH(n+1)F bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1).

The C(n)V register stores the value of FTM counter when the selected edge by channel (n) is detected at channel (n) input. The C(n+1)V register stores the value of FTM counter when the selected edge by channel (n+1) is detected at channel (n) input.

In this mode, a coherency mechanism ensures coherent data when the C(n)V and C(n +1)V registers are read. The only requirement is that C(n)V must be read before C(n +1)V.

Note

- The CH(n)F, CH(n)IE, MS(n)A, ELS(n)B, and ELS(n)A bits are channel (n) bits.
- The CH(n+1)F, CH(n+1)IE, MS(n+1)A, ELS(n+1)B, and ELS(n+1)A bits are channel (n+1) bits.
- The Dual Edge Capture mode must be used with ELS(n)B:ELS(n)A = 0:1 or 1:0, ELS(n+1)B:ELS(n+1)A = 0:1 or 1:0 and the FTM counter in Free running counter.

26.4.23.1 One-Shot Capture mode

The One-Shot Capture mode is selected when (FTMEN = 1), (DECAPEN = 1), and (MS(n)A = 0). In this capture mode, only one pair of edges at the channel (n) input is captured. The ELS(n)B:ELS(n)A bits select the first edge to be captured, and ELS(n +1)B:ELS(n+1)A bits select the second edge to be captured.

The edge captures are enabled while DECAP bit is set. For each new measurement in One-Shot Capture mode, first the CH(n)F and CH(n+1) bits must be cleared, and then the DECAP bit must be set.

In this mode, the DECAP bit is automatically cleared by FTM when the edge selected by channel (n+1) is captured. Therefore, while DECAP bit is set, the one-shot capture is in process. When this bit is cleared, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

Similarly, when the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

26.4.23.2 Continuous Capture mode

The Continuous Capture mode is selected when (FTMEN = 1), (DECAPEN = 1), and (MS(n)A = 1). In this capture mode, the edges at the channel (n) input are captured continuously. The ELS(n)B:ELS(n)A bits select the initial edge to be captured, and ELS(n+1)B:ELS(n+1)A bits select the final edge to be captured.

The edge captures are enabled while DECAP bit is set. For the initial use, first the CH(n)F and CH(n+1)F bits must be cleared, and then DECAP bit must be set to start the continuous measurements.

When the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers. The latest captured values are always available in these registers even after the DECAP bit is cleared.

In this mode, it is possible to clear only the CH(n+1)F bit. Therefore, when the CH(n+1)F bit is set again, the latest captured values are available in C(n)V and C(n+1)V registers.

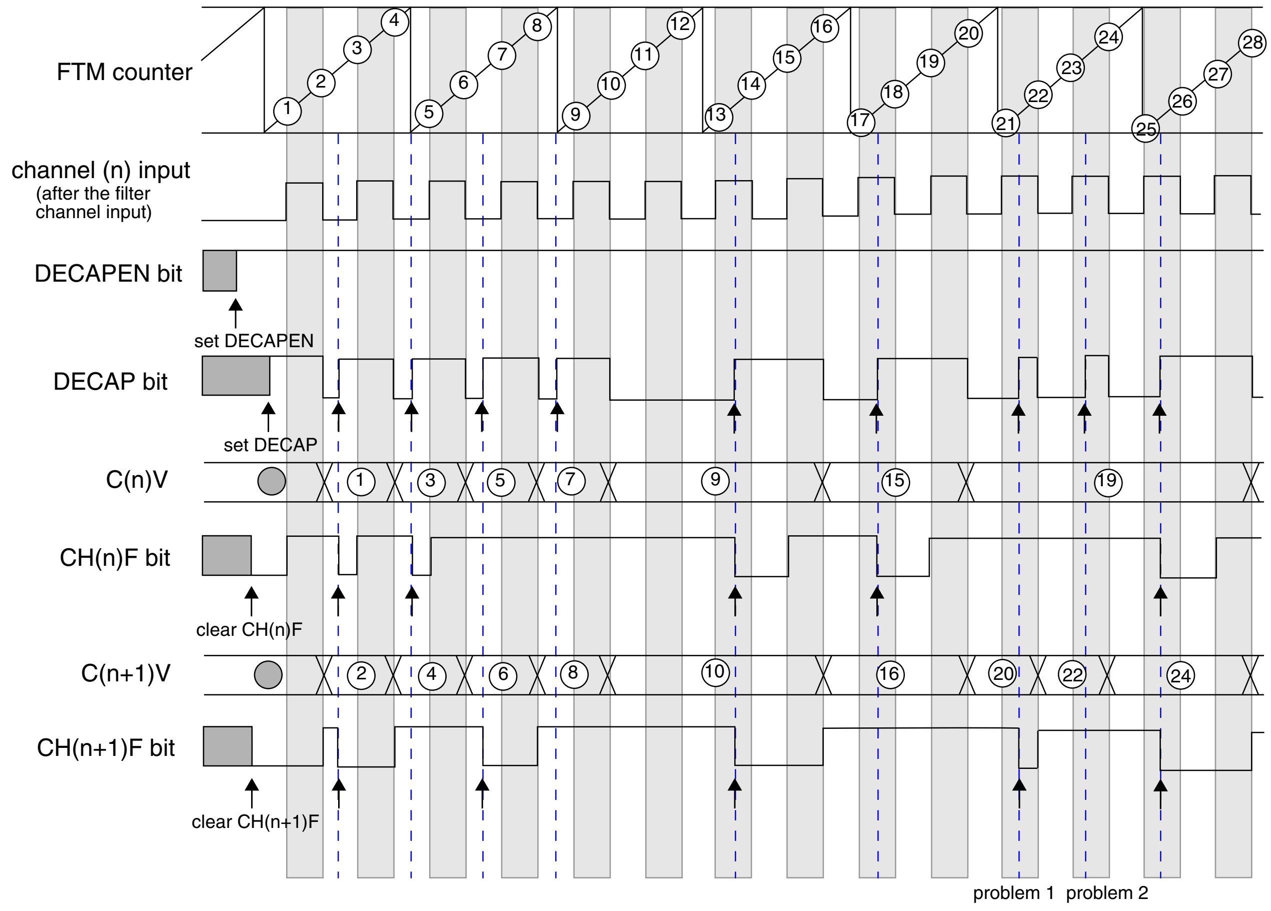
For a new sequence of the measurements in the Dual Edge Capture – Continuous mode, clear the CH(n)F and CH(n+1)F bits to start new measurements.

26.4.23.3 Pulse width measurement

If the channel (n) is configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1) and the channel (n+1) to capture falling edges (ELS(n+1)B:ELS(n+1)A = 1:0), then the positive polarity pulse width is measured. If the channel (n) is configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0) and the channel (n+1) to capture rising edges (ELS(n +1)B:ELS(n+1)A = 0:1), then the negative polarity pulse width is measured.

The pulse width measurement can be made in One-Shot Capture mode or Continuous Capture mode.

The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the measurement of next positive polarity pulse width. The CH(n)F bit is set when the first edge of this pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second edge of this pulse is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 26-201. Dual Edge Capture – One-Shot mode for positive polarity pulse width measurement

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first edge of the positive polarity pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set when the second edge of this pulse is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. The CH(n+1)F bit indicates when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.

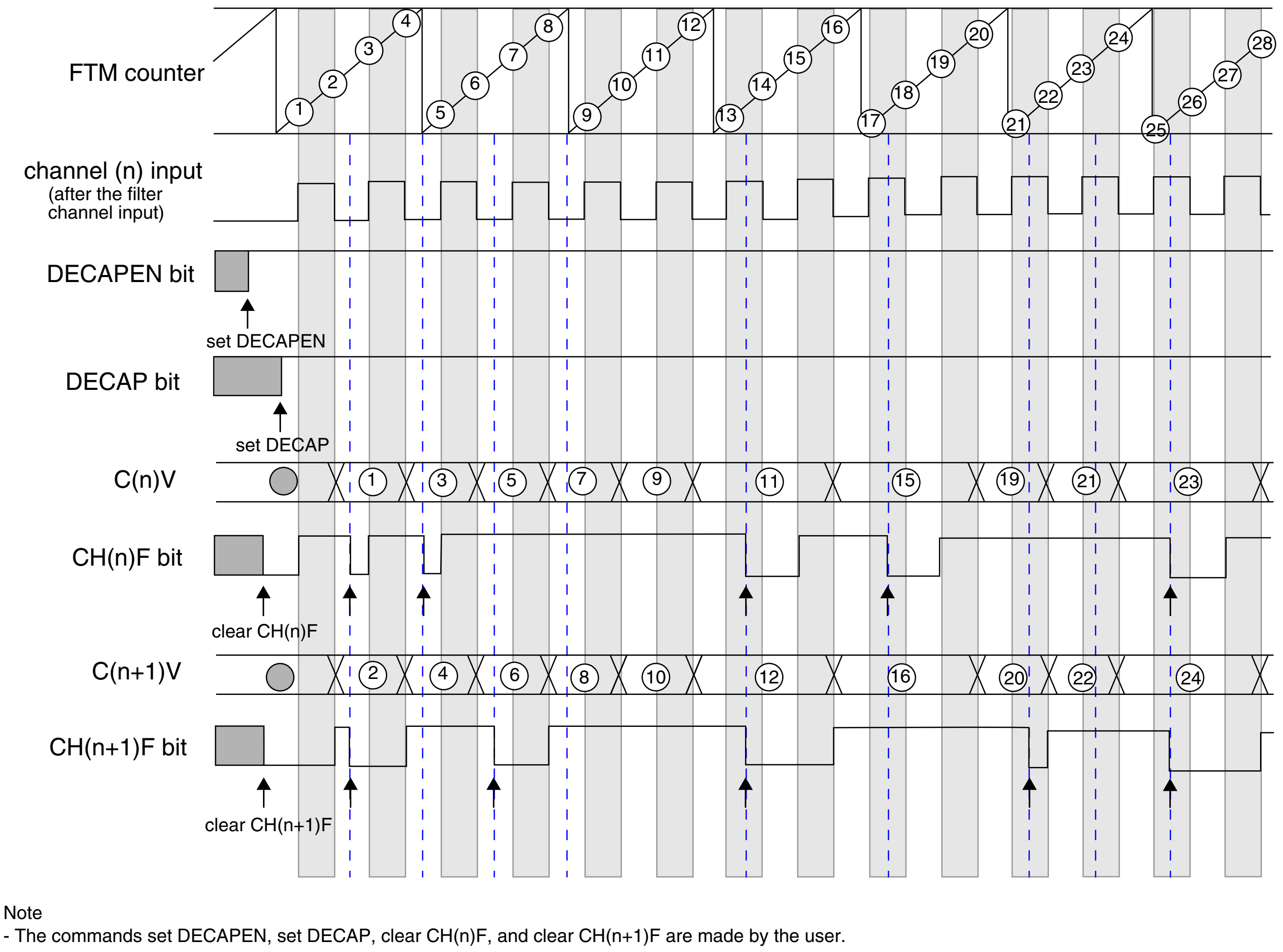


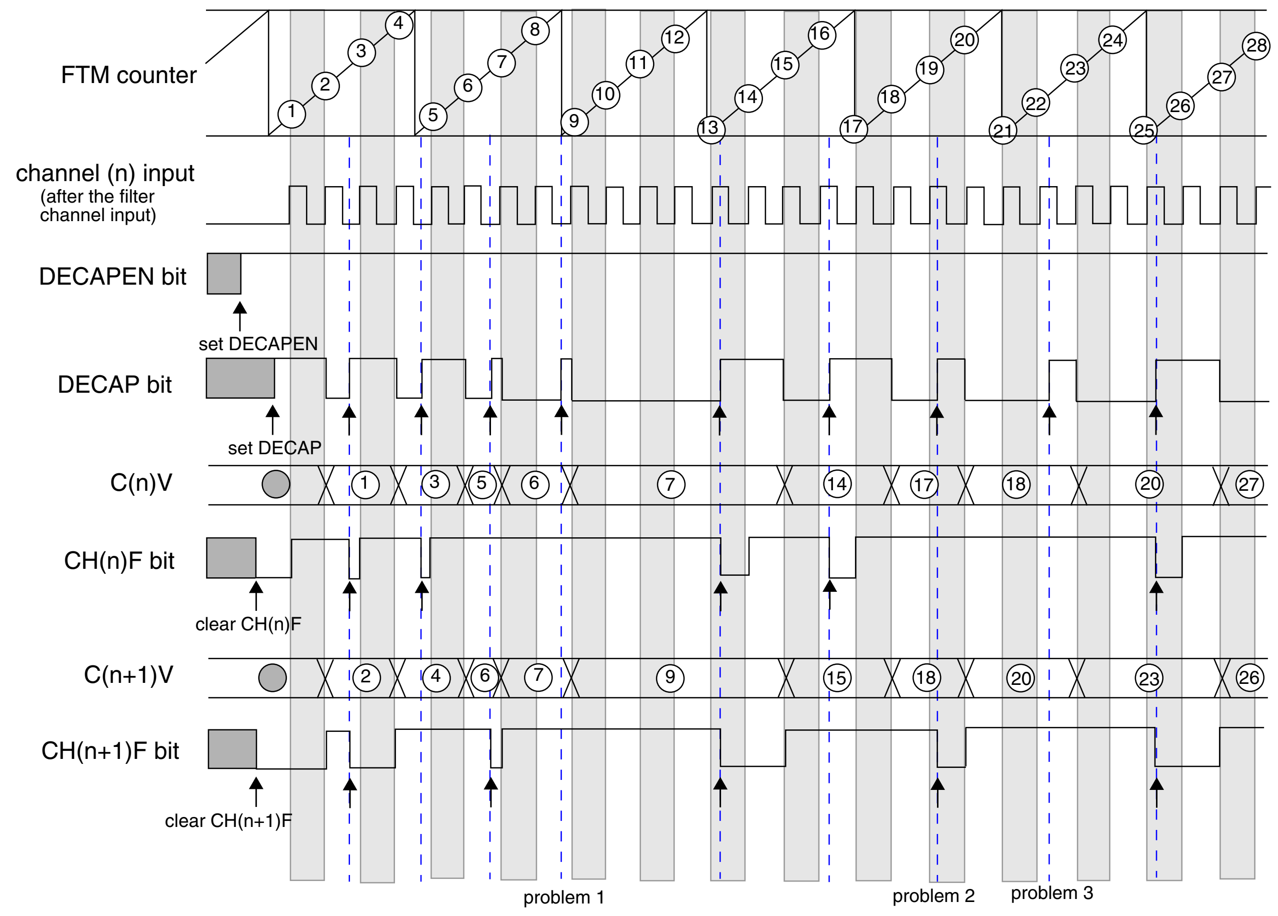
Figure 26-202. Dual Edge Capture – Continuous mode for positive polarity pulse width measurement

26.4.23.4 Period measurement

If the channels (n) and (n+1) are configured to capture consecutive edges of the same polarity, then the period of the channel (n) input signal is measured. If both channels (n) and (n+1) are configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1 and ELS(n +1)B:ELS(n+1)A = 0:1), then the period between two consecutive rising edges is measured. If both channels (n) and (n+1) are configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0 and ELS(n+1)B:ELS(n+1)A = 1:0), then the period between two consecutive falling edges is measured.

The period measurement can be made in One-Shot Capture mode or Continuous Capture mode.

The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the measurement of next period. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second rising edge is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two selected edges were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 0, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 3: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 26-203. Dual Edge Capture – One-Shot mode to measure of the period between two consecutive rising edges

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set when the second rising edge is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. The CH(n+1)F bit indicates when two edges of the period were captured and the C(n)V and C(n+1)V registers are ready for reading.

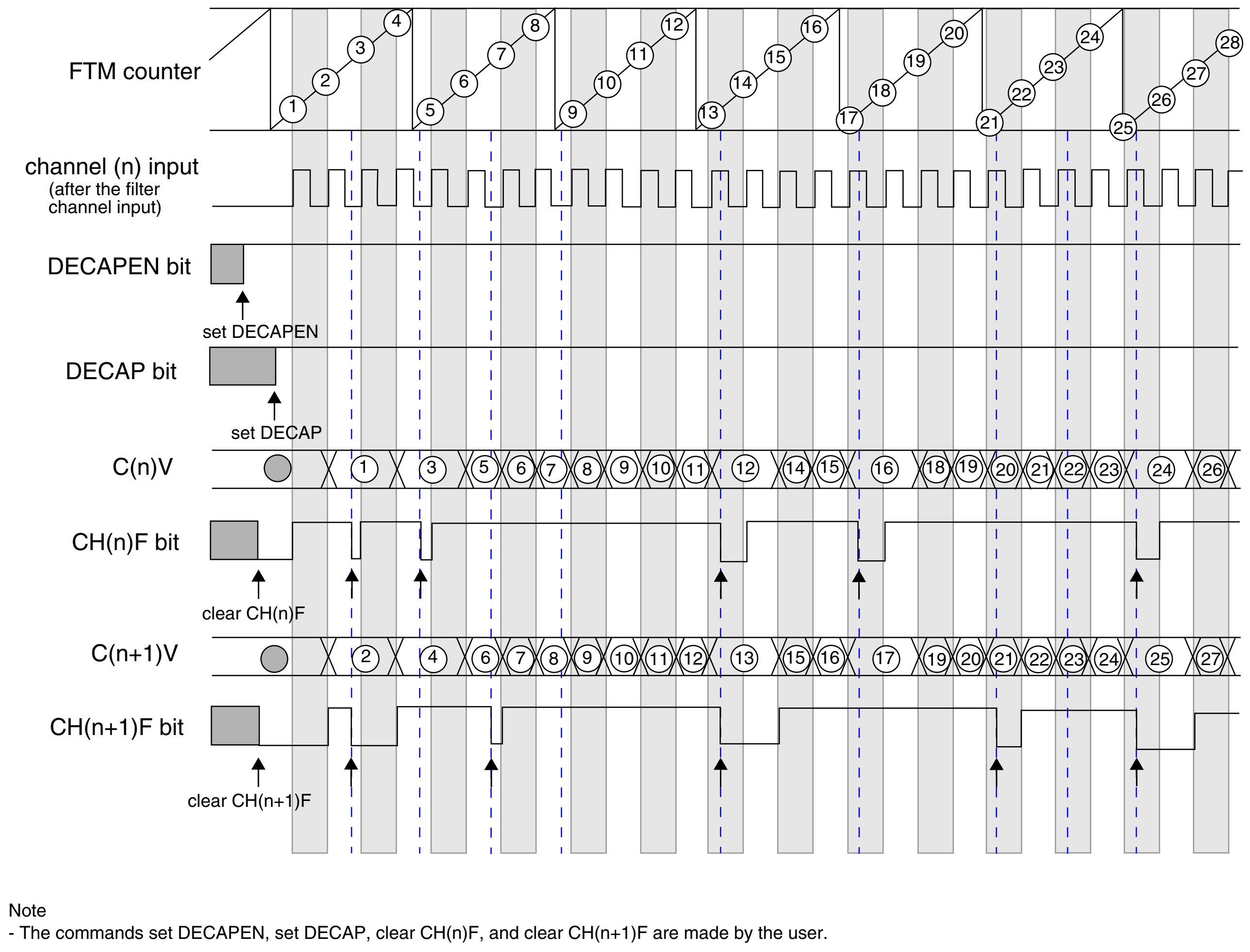


Figure 26-204. Dual Edge Capture – Continuous mode to measure of the period between two consecutive rising edges

26.4.23.5 Read coherency mechanism

The Dual Edge Capture mode implements a read coherency mechanism between the FTM counter value captured in C(n)V and C(n+1)V registers. The read coherency mechanism is illustrated in the following figure. In this example, the channels (n) and (n

+1) are in Dual Edge Capture – Continuous mode for positive polarity pulse width measurement. Thus, the channel (n) is configured to capture the FTM counter value when there is a rising edge at channel (n) input signal, and channel (n+1) to capture the FTM counter value when there is a falling edge at channel (n) input signal.

When a rising edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n) capture buffer. The channel (n) capture buffer value is transferred to C(n)V register when a falling edge occurs in the channel (n) input signal. C(n)V register has the FTM counter value when the previous rising edge occurred, and the channel (n) capture buffer has the FTM counter value when the last rising edge occurred.

When a falling edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n+1) capture buffer. The channel (n+1) capture buffer value is transferred to C(n+1)V register when the C(n)V register is read.

In the following figure, the read of C(n)V returns the FTM counter value when the event 1 occurred and the read of C(n+1)V returns the FTM counter value when the event 2 occurred.

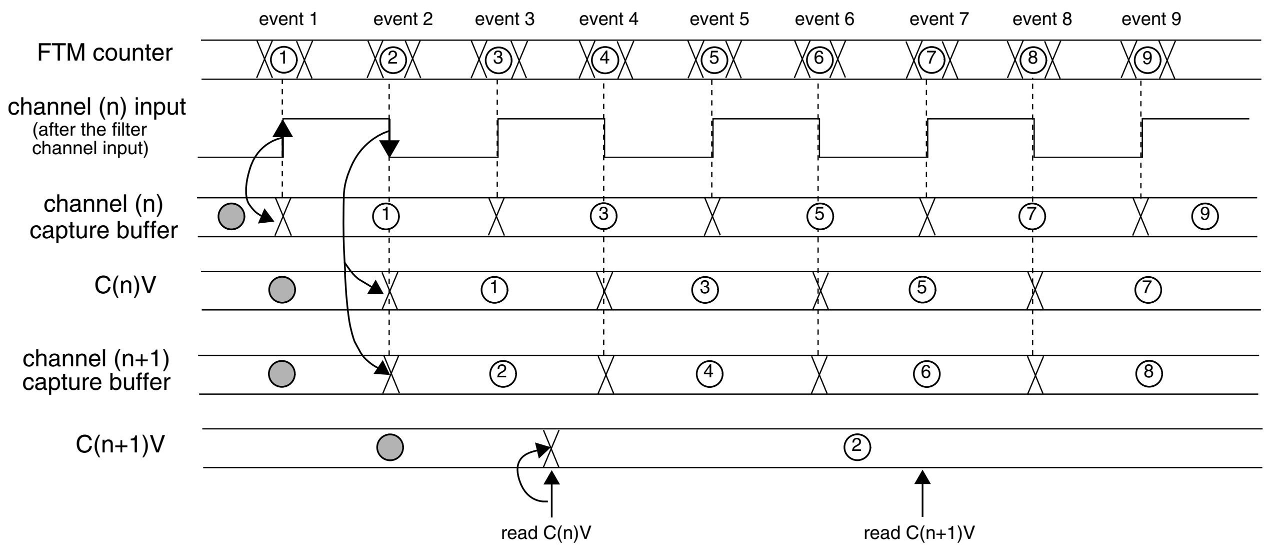


Figure 26-205. Dual Edge Capture mode read coherency mechanism

C(n)V register must be read prior to C(n+1)V register in dual edge capture one-shot and continuous modes for the read coherency mechanism works properly.

26.4.24 Debug mode

When the chip is in Debug mode, the BDMMODE[1:0] bits select the behavior of the FTM counter, the CH(n)F bit, the channels output, and the writes to the MOD, CNTIN, and C(n)V registers according to the following table.

Table 26-188. FTM behavior when the chip Is in Debug mode

BDMMODE	FTM Counter	CH(n)F Bit	FTM Channels Output	Writes to MOD, CNTIN, and C(n)V Registers
00	Stopped	can be set	Functional mode	Writes to these registers bypass the registers buffers
01	Stopped	is not set	The channels outputs are forced to their safe value according to POLn bit	Writes to these registers bypass the registers buffers
10	Stopped	is not set	The channels outputs are frozen when the chip enters in Debug mode	Writes to these registers bypass the registers buffers
11	Functional mode	can be set	Functional mode	Functional mode

Note that if BDMMODE[1:0] = 2'b00 then the channels outputs remain at the value when the chip enters in Debug mode, because the FTM counter is stopped. However, the following situations modify the channels outputs in this Debug mode.

- Write any value to CNT register; see Counter reset. In this case, the FTM counter is updated with the CNTIN register value and the channels outputs are updated to the initial value – except for those channels set to Output Compare mode.
- FTM counter is reset by PWM Synchronization mode; see FTM counter synchronization. In this case, the FTM counter is updated with the CNTIN register value and the channels outputs are updated to the initial value – except for channels in Output Compare mode.
- In the channels outputs initialization, the channel (n) output is forced to the CH(n)OI bit value when the value 1 is written to INIT bit. See Initialization.

Note

The BDMMODE[1:0] = 2'b00 must not be used with the Fault control. Even if the fault control is enabled and a fault condition exists, the channels outputs values are updated as above.

26.4.25 Intermediate load

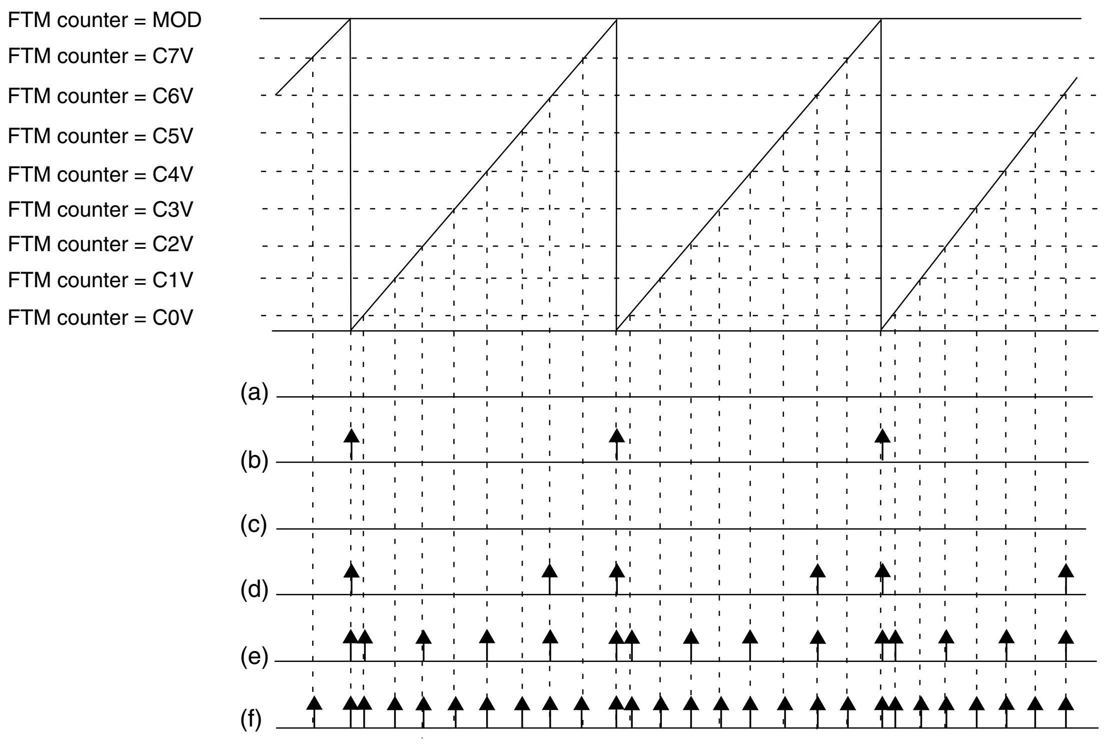
The PWMLOAD register allows to update the MOD, CNTIN, and C(n)V registers with the content of the register buffer at a defined load point. In this case, it is not required to use the PWM synchronization.

There are multiple possible loading points for intermediate load:

Table 26-189. When possible loading points are enabled

Loading point	Enabled
When the FTM counter wraps from MOD value to CNTIN value	Always
At the channel (j) match (FTM counter = C(j)V)	When CHjSEL = 1

The following figure shows some examples of enabled loading points.



NOTE

(a) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(b) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(c) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 1, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(d) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(e) LDOK = 1, CH0SEL = 1, CH1SEL = 0, CH2SEL = 1, CH3SEL = 0, CH4SEL = 1, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(f) LDOK = 1, CH0SEL = 1, CH1SEL = 1, CH2SEL = 1, CH3SEL = 1, CH4SEL = 1, CH5SEL = 1, CH6SEL = 1, CH7SEL = 1

Figure 26-206. Loading points for intermediate load

After enabling the loading points, the LDOK bit must be set for the load to occur. In this case, the load occurs at the next enabled loading point according to the following conditions:

Table 26-190. Conditions for loads occurring at the next enabled loading point

When a new value was written	Then
To the MOD register	The MOD register is updated with its write buffer value.

Table continues on the next page...

Table 26-190. Conditions for loads occurring at the next enabled loading point (continued)

When a new value was written	Then
To the CNTIN register and CNTINC = 1	The CNTIN register is updated with its write buffer value.
To the C(n)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1)	The C(n)V register is updated with its write buffer value.
To the C(n+1)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1)	The C(n+1)V register is updated with its write buffer value.

NOTE

- If ELSjB and ELSjA bits are different from zero, then the channel (j) output signal is generated according to the configured output mode. If ELSjB and ELSjA bits are zero, then the generated signal is not available on channel (j) output.
- If CHjIE = 1, then the channel (j) interrupt is generated when the channel (j) match occurs.
- At the intermediate load neither the channels outputs nor the FTM counter are changed. Software must set the intermediate load at a safe point in time.
- The intermediate load feature must be used only in Combine mode.

26.4.26 Global time base (GTB)

The global time base (GTB) is a FTM function that allows the synchronization of multiple FTM modules on a chip. The following figure shows an example of the GTB feature used to synchronize two FTM modules. In this case, the FTM A and B channels can behave as if just one FTM module was used, that is, a global time base.

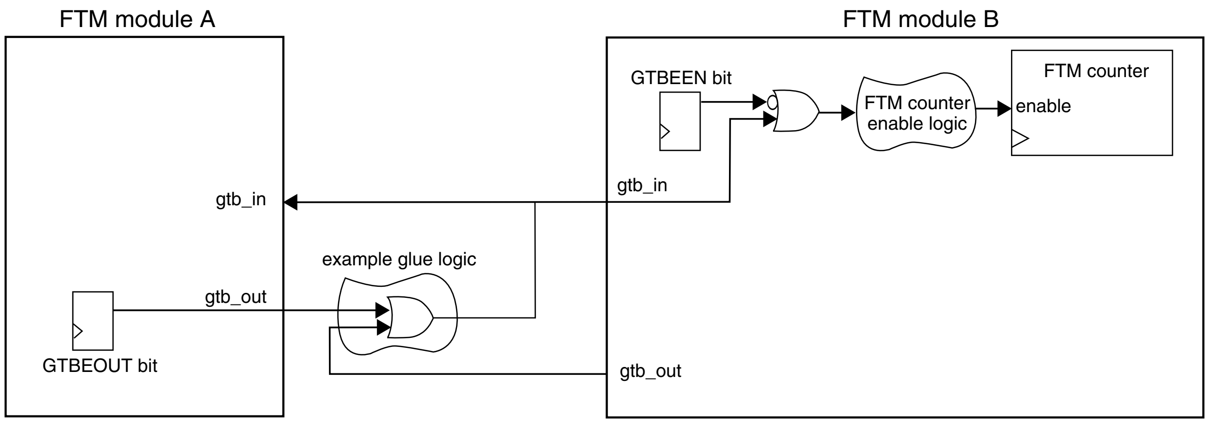


Figure 26-207. Global time base (GTB) block diagram

The GTB functionality is implemented by the GTBEEN and GTBEOUT bits in the CONF register, the input signal *gtb_in*, and the output signal *gtb_out*. The GTBEEN bit enables gtb_in to control the FTM counter enable signal:

- If GTBEEN = 0, each one of FTM modules works independently according to their configured mode.
- If GTBEEN = 1, the FTM counter update is enabled only when gtb_in is 1.

In the configuration described in the preceding figure, FTM modules A and B have their FTM counters enabled if at least one of the gtb_out signals from one of the FTM modules is 1. There are several possible configurations for the interconnection of the gtb_in and gtb_out signals, represented by the example glue logic shown in the figure. Note that these configurations are chip-dependent and implemented outside of the FTM modules. See the chip configuration details for the chip's specific implementation.

NOTE

- In order to use the GTB signals to synchronize the FTM counter of different FTM modules, the configuration of each FTM module should guarantee that its FTM counter starts counting as soon as the gtb_in signal is 1.
- The GTB feature does not provide continuous synchronization of FTM counters, meaning that the FTM counters may lose synchronization during FTM operation. The GTB feature only allows the FTM counters to *start* their operation synchronously.

26.4.26.1 Enabling the global time base (GTB)

To enable the GTB feature, follow these steps for each participating FTM module:

1. Stop the FTM counter: Write 00b to SC[CLKS].
2. Program the FTM to the intended configuration. The FTM counter mode needs to be consistent across all participating modules.
3. Write 1 to CONF[GTBEEN] and write 0 to CONF[GTBEOUT] at the same time.
4. Select the intended FTM counter clock source in SC[CLKS]. The clock source needs to be consistent across all participating modules.
5. Reset the FTM counter: Write any value to the CNT register.

To initiate the GTB feature in the configuration described in the preceding figure, write 1 to CONF[GTBEOUT] in the FTM module used as the time base.

26.5 Reset overview

The FTM is reset whenever any chip reset occurs.

When the FTM exits from reset:

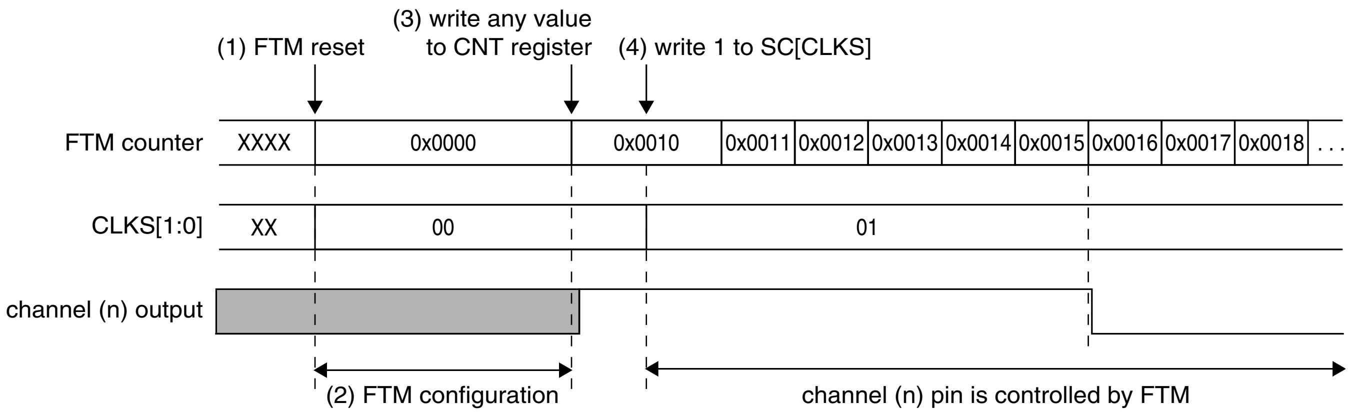
- the FTM counter and the prescaler counter are zero and are stopped (CLKS[1:0] = 00b);
- the timer overflow interrupt is zero, see Timer Overflow Interrupt;
- the channels interrupts are zero, see Channel (n) Interrupt;
- the fault interrupt is zero, see Fault Interrupt;
- the channels are in input capture mode, see Input Capture mode;
- the channels outputs are zero;
- the channels pins are not controlled by FTM (ELS(n)B:ELS(n)A = 0:0) (See the table in the description of CnSC register).

The following figure shows the FTM behavior after the reset. At the reset (item 1), the FTM counter is disabled (see the description of the CLKS field in the Status and Control register), its value is updated to zero and the pins are not controlled by FTM (See the table in the description of CnSC register).

After the reset, the FTM should be configurated (item 2). It is necessary to define the FTM counter mode, the FTM counting limits (MOD and CNTIN registers value), the channels mode and CnV registers value according to the channels mode.

Thus, it is recommended to write any value to CNT register (item 3). This write updates the FTM counter with the CNTIN register value and the channels output with its initial value (except for channels in output compare mode) (Counter reset).

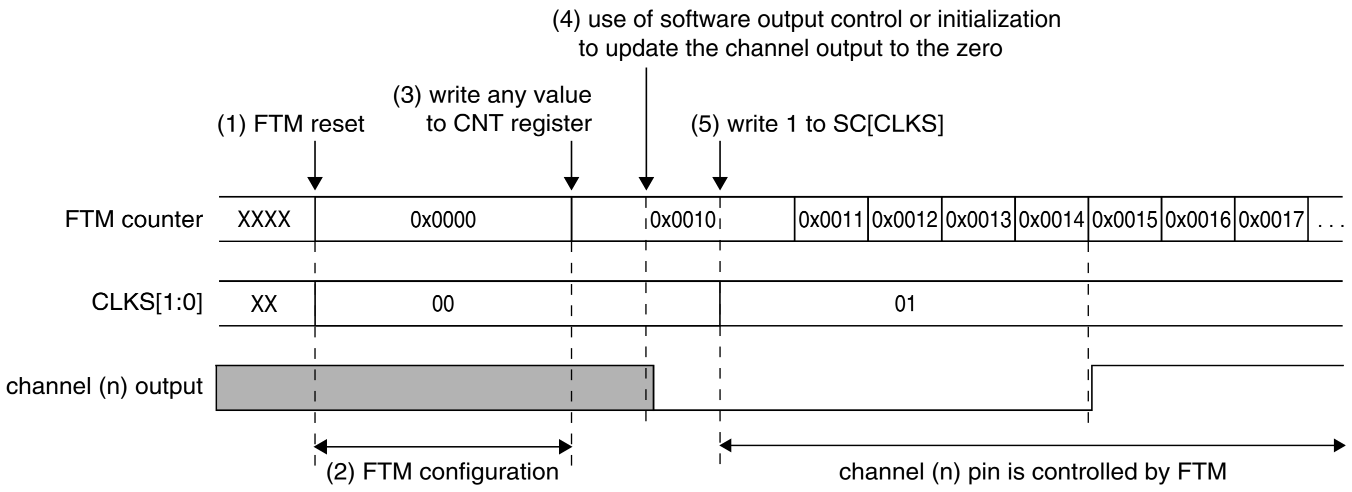
The next step is to select the FTM counter clock by the CLKS[1:0] bits (item 4). It is important to highlight that the pins are only controlled by FTM when CLKS[1:0] bits are different from zero (See the table in the description of CnSC register).



NOTES:
- CNTIN = 0x0010
- Channel (n) is in low-true combine mode with CNTIN < C(n)V < C(n+1)V < MOD
- C(n)V = 0x0015

Figure 26-208. FTM behavior after reset when the channel (n) is in Combine mode

The following figure shows an example when the channel (n) is in Output Compare mode and the channel (n) output is toggled when there is a match. In the Output Compare mode, the channel output is not updated to its initial value when there is a write to CNT register (item 3). In this case, use the software output control (Software output control) or the initialization (Initialization) to update the channel output to the selected value (item 4).


(4) use of software output control or initialization to update the channel output to the zero
(1) FTM reset
(3) write any value to CNT register
(5) write 1 to SC[CLKS]
FTM counter
XXXX
0x0000
0x0010
0x0011
0x0012
0x0013
0x0014
0x0015
0x0016
0x0017
. . .
CLKS[1:0]
XX
00
01
channel (n) output
(2) FTM configuration
channel (n) pin is controlled by FTM

NOTES:
- CNTIN = 0x0010
- Channel (n) is in output compare and the channel (n) output is toggled when there is a match
- C(n)V = 0x0014

Figure 26-209. FTM behavior after reset when the channel (n) is in Output Compare mode

26.6 FTM Interrupts

26.6.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

26.6.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

26.6.3 Fault Interrupt

The fault interrupt is generated when (FAULTIE = 1) and (FAULTF = 1).

Chapter 27
Pulse Width Timer (PWT)

27.1 Introduction

27.1.1 Features

The pulse width timer (PWT) includes the following features:

- Automatic measurement of pulse width with 16-bit resolution
- Separate positive and negative pulse width measurements
- Programmable triggering edge for starting measurement
- Programmable measuring time between successive alternating edges, rising edges or falling edges
- Programmable prescaler from clock input as 16-bit counter time base
- Two selectable clock sources—bus clock and alternative clock
- Four selectable pulse inputs
- Programmable interrupt generation upon pulse width value updated and counter overflow

27.1.2 Modes of operation

The following table describes the operation of the PWT module in various modes.

Modes	Description
Run	When enabled, the pulse width timer module is active.
Wait	When enabled, the pulse width timer module is active and can perform the waking up function if the corresponding interrupt is enabled.
Stop	The pulse width timer module is halted when entering stop and the register contents and operating status is preserved. If stop exits with reset then the module resets. If stop exits with another source, the module resumes operation based on module status upon exit.
Active background	Upon entering Debug mode, the PWT suspends all counting and pulse edge detection until the microcontroller returns to normal user operating mode. Counting and edge detection resume from the suspended value when normal user operating mode returns as long as the PWTSR bit (PWT software reset) is not written to 1 and the PWT module is still enabled.

27.1.3 Block diagram

The following is the block diagram of the pulse width timer module (PWT).

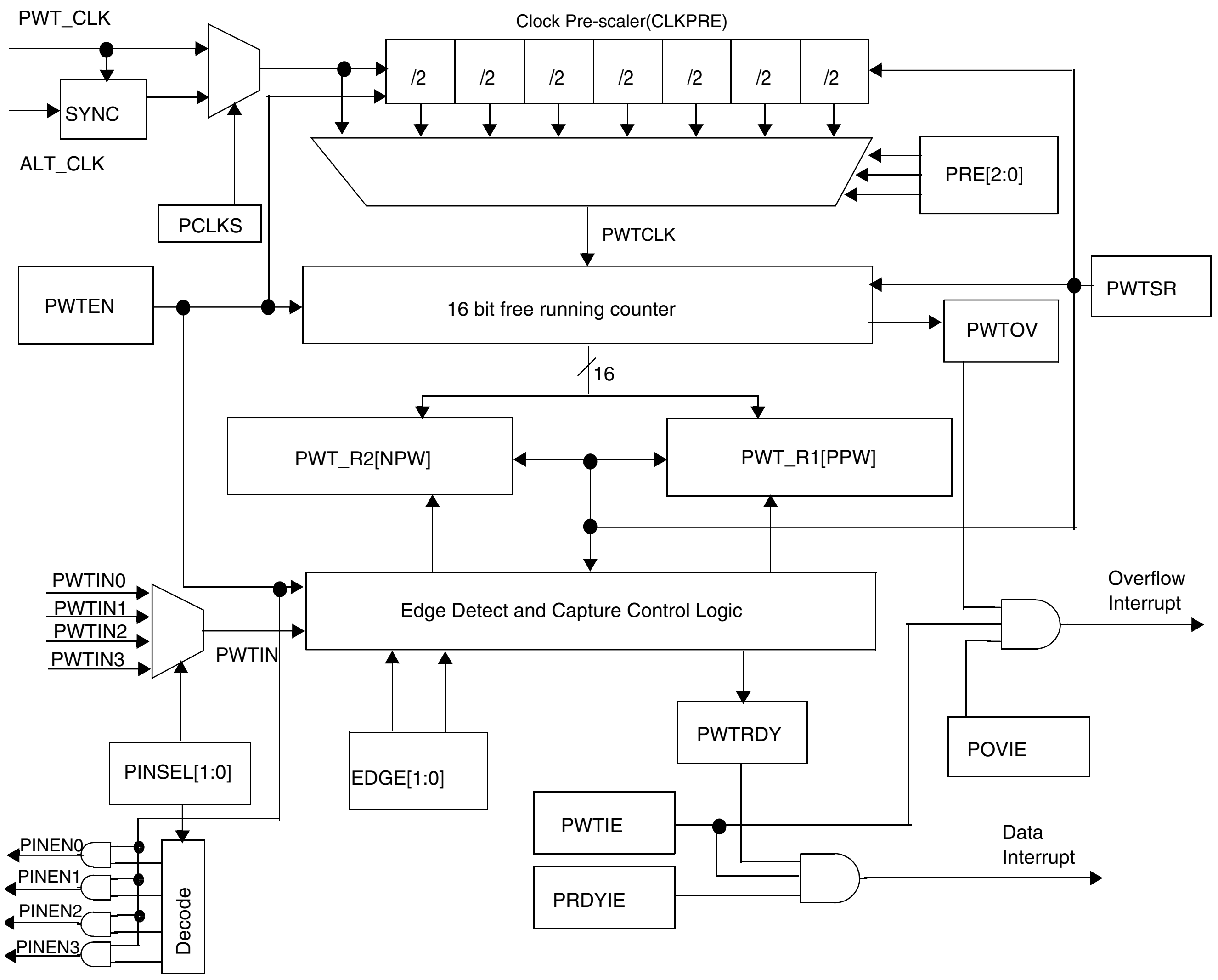


Figure 27-1. Pulse width timer (PWT) block diagram

NOTE

The PWT_CLK depends on the Chip input clock. For this chip it is TIMER_CLK.

27.2 PWT signal description

Table 27-2. PWT signal description

Signal	I/O	Pullup	Description
PWTIN[3:0]	I	No	Pulse Inputs
ALTCLK	I	No	Alternative clock source for the counter

27.2.1 PWTIN[3:0] - Pulse Width Timer Capture Inputs

The input signals are pulse capture inputs which can come from internal or external sources. The PWT input is selected by PINSEL[1:0] to be routed to the pulse width timer. If the input comes from external source and is selected as the PWT input, the input port is enabled for PWT function by PINSEL[1:0] automatically. The minimum pulse width to be measured is 1 PWTCLK cycle, any pulse narrower than this value is ignored by PWT module. The PWTCLK cycle time depends on the PWT clock source selection and prescaler rate setting.

27.2.2 ALTCLK- Alternative Clock Source for Counter

The PWT has an alternative clock input ALTCLK which can be selected as the clock source of the counter when R1[PCLKS] is set. The ALTCLK input must be synchronized by the bus clock. Variations in duty cycle and clock jitter must also be accommodated so that the ALTCLK signal must not exceed one-fourth of the bus frequency. The ALTCLK pin can be shared with a general-purpose port pin. See the Pins and Connections chapter for the pin location and priority of this function.

27.3 Memory Map and Register Descriptions

PWT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_3000	Pulse Width Timer Register 1 (PWT_R1)	32	R/W	0000_0000h	27.3.1/468
4003_3004	Pulse Width Timer Register 2 (PWT_R2)	32	R	0000_0000h	27.3.2/470

27.3.1 Pulse Width Timer Register 1 (PWT_R1)

This register defines the general control and status bits for the PWT. It also contains the Positive Pulse Width contents.

Address: 4003_3000h base + 0h offset = 4003_3000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PPW															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PCLKS	PINSEL		EDGE		PRE			PWTEN	PWTIE	PRDYIE	POVIE	0	0	PWTRDY	PWTOV
W													PWTSR	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWT_R1 field descriptions

Field	Description
31–16 PPW	Positive Pulse Width Captured positive pulse width value. It is suggested to use half-word (16-bit) or word (32-bit) to read out this value.
15 PCLKS	PWT Clock Source Selection Controls the selection of clock source for the PWT counter. 0 TIMER_CLK is selected as the clock source of PWT counter. 1 Alternative clock is selected as the clock source of PWT counter.
14–13 PINSEL	PWT Pulse Inputs Selection Enables the corresponding PWT input port, if this PWT input comes from an external source. 00 PWTIN[0] is enabled. 01 PWTIN[1] is enabled.

Table continues on the next page...

PWT_R1 field descriptions (continued)

Field	Description
	10 PWTIN[2] enabled. 11 PWTIN[3] enabled.
12–11 EDGE	PWT Input Edge Sensitivity Selects which edge triggers the pulse width measurement and which edges trigger the capture. If user needs to change the trigger and capture mode by changing the value of EDGE[1:0], a PWT software reset is required after changing the EDGE[1:0] value. Clearing PWTEN and then setting it has the same effect. 00 The first falling-edge starts the pulse width measurement, and on all the subsequent falling edges, the pulse width is captured. 01 The first rising edge starts the pulse width measurement, and on all the subsequent rising and falling edges, the pulse width is captured. 10 The first falling edge starts the pulse width measurement, and on all the subsequent rising and falling edges, the pulse width is captured. 11 The first-rising edge starts the pulse width measurement, and on all the subsequent rising edges, the pulse width is captured.
10–8 PRE	PWT Clock Prescaler (CLKPRE) Setting Selects the value by which the clock is divided to clock the PWT counter. 000 Clock divided by 1. 001 Clock divided by 2. 010 Clock divided by 4. 011 Clock divided by 8. 100 Clock divided by 16. 101 Clock divided by 32. 110 Clock divided by 64. 111 Clock divided by 128.
7 PWTEN	PWT Module Enable Enables/disables the PWT module. To avoid unexpected behavior, do not change any PWT configurations as long as PWTEN is set. 0 The PWT is disabled. 1 The PWT is enabled.
6 PWTIE	PWT Module Interrupt Enable Enables the PWT module to generate an interrupt. 0 Disables the PWT to generate interrupt. 1 Enables the PWT to generate interrupt.
5 PRDYIE	PWT Pulse Width Data Ready Interrupt Enable Enables/disables the PWT to generate an interrupt when PWTRDY is set as long as PWTIE is set. 0 Disable PWT to generate interrupt when PWTRDY is set. 1 Enable PWT to generate interrupt when PWTRDY is set.
4 POVIE	PWT Counter Overflow Interrupt Enable Enables/disables the PWT to generate an interrupt when PWTOV is set due to PWT counter overflow.

Table continues on the next page...

PWT_R1 field descriptions (continued)

Field	Description
	0 Disable PWT to generate interrupt when PWTOV is set. 1 Enable PWT to generate interrupt when PWTOV is set.
3 PWTSR	PWT Soft Reset Performs a soft reset to the PWT. This field always reads as 0. 0 No action taken. 1 Writing 1 to this field will perform soft reset to PWT.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 PWTRDY	PWT Pulse Width Valid Indicates that the PWT Pulse Width register(s) has been updated and is ready to be read. This field is cleared by reading PWTRDY and then writing 0 to PWTRDY bit when PWTRDY is set. Writing 1 to this field has no effect. PWTRDY setting is associated with the EDGE[1:0] bits. 0 PWT pulse width register(s) is not up-to-date. 1 PWT pulse width register(s) has been updated.
0 PWTOV	PWT Counter Overflow Indicates that the PWT counter has run from 0x0000_0xFFFF to 0x0000_0x0000. This field is cleared by writing 0 to PWTOV when PWTOV is set. Writing 1 to this field has no effect. If another overflow occurs when this field is being cleared, the clearing fails. 0 PWT counter no overflow. 1 PWT counter runs from 0xFFFF to 0x0000.

27.3.2 Pulse Width Timer Register 2 (PWT_R2)

This register contains the Negative Pulse Width contents and the 16-bit free running counter contents.

Address: 4003_3000h base + 4h offset = 4003_3004h

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

PWT_R2 field descriptions

Field	Description
31–16 PWTC	PWT Counter. It is suggested to use half-word (16-bit) or word (32-bit) to read out this value.
NPW	Negative Pulse Width. It is suggested to use half-word (16-bit) or word (32-bit) to read out this value.

27.4 Functional Description

27.4.1 PWT Counter and PWT Clock Prescaler

The pulse width timer (PWT) measures duration of a pulse or the period of a signal input to the PWTIN by a 16-bit free running counter (PWT_R2[PWTC]). There is a clock prescaler (CLKPRE) in PWT module that provides the frequency divided clock to PWT_R2[PWTC]. The clock prescaler can select clock input from bus clock and alternative clock by PWT_R1[PCKLS].

The PWT counter uses the frequency divided clock from CLKPRE for counter advancing. The frequency of prescaler is programmable as the clock frequency divided by 1, 2, 4, 8, 16, 32, 64, 128 (depending on the setting of PRE[2:0]).

As soon as the PWT counter is enabled, it starts counting using the selected and divided clock source; the counter is cleared without loading to the registers when the first valid edge (trigger edge) is detected. If no valid trigger edge is detected for a long time, it is possible for the counter to overflow. When 16-bit free running counter is running, any edge to be measured after the trigger edge causes the value of PWT_R2[PWTC] to be uploaded to the appropriate pulse width registers. At the same time, PWT_R2[PWTC] will be reset to 0x0000 and the clock prescaler output will also be reset together. PWT_R2[PWTC] will then start advancing again with the input clock. If the PWT_R2[PWTC] runs from 0xFFFF to 0x0000, the PWTOV bit is set.

27.4.2 Edge detection and capture control

The edge detection and capture control part detects measurement trigger edges and controls when and which pulse width register(s) will be updated.

Based on the setting of EDGE[1:0], the edge detection logic determines the starting and ending edge of the pulse width to be measured on PWTIN, and the registers fields to be updated. See Edge detection and capture control for details.

The PWTIN can be selected from one of four sources by configuring PINSEL[1:0].

It must be noted that inside the edge detection and capture logic, the system slave clock is used for PWT to sample and synchronize the PWTIN pulse, therefore the minimal PWTIN pulse width is determined by this slave clock frequency. For example, if this clock frequency is 50 MHz, the minimal PWTIN pulse width must be longer than 20 ns(the period of 50 MHz), otherwise, PWT won't be able to capture this pulse and the counter would overflow. See the chip configuration chapters to check the slave clock frequency to PWT. Also, if there isn't any valid edge of the PWTIN width for a long time, the PWT counter would overflow as well.

When EDGE[1:0] is 00, the first falling edge is the trigger edge from which the pulse width begins to be measured. The counter value is uploaded to PWT_R1[PPW] upon each of the subsequent falling edges. When EDGE[1:0] is 11, the first rising edge is the trigger edge from which the pulse width begins to be measured. The counter value is uploaded to PWT_R2[NPW] upon each successive rising-edge. In these two cases, the period of PWTIN is measured.

When EDGE[1:0] is 01, the first rising edge is the trigger edge. The pulse width begins to be measured from this edge. PWT_R1[PPW] is uploaded upon each of the subsequent falling edges. PWT_R2[NPW] is uploaded upon each successive rising edge. When EDGE[1:0] is 10, the first falling edge is the trigger edge from which the pulse width is measured. PWT_R2[NPW] is uploaded on each successive rising edge and PWT_R1[PPW] is uploaded on each successive falling edge. In these two cases, the positive pulse and negative pulse are measured separately and the positive pulse width is uploaded into PWT_R1[PPW]. The negative pulse width is uploaded into PWT_R2[NPW].

The following figure illustrates the trigger edge detection and pulse width registers update of PWT.

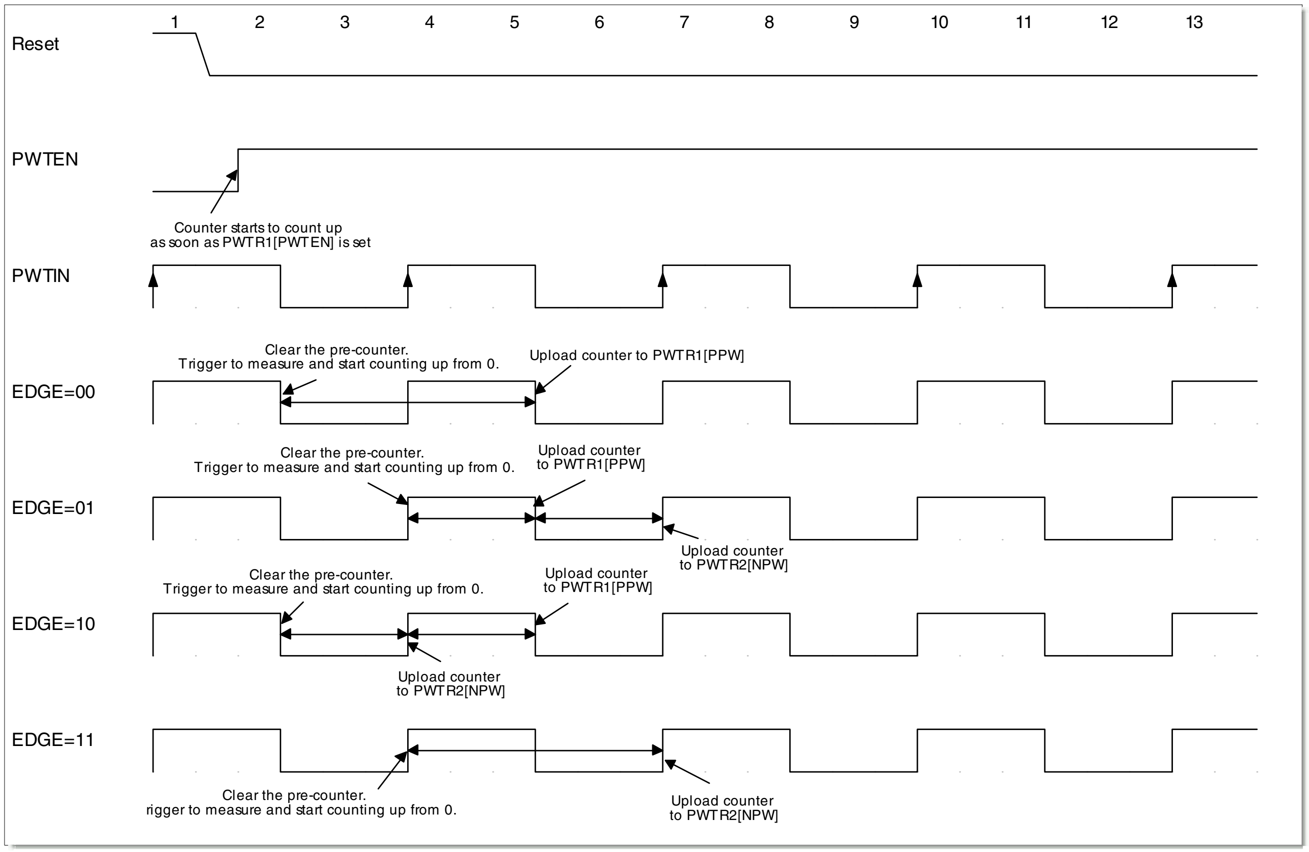

1
2
3
4
5
6
7
8
9
10
11
12
13
Reset
PWTEN
Counter starts to count up as soon as PWTR1[PWTEN] is set
PWTIN
Clear the pre-counter. Trigger to measure and start counting up from 0.
Upload counter to PWTR1[PPW]
EDGE=00
Clear the pre-counter. Trigger to measure and start counting up from 0.
Upload counter to PWTR1[PPW]
EDGE=01
Upload counter to PWTR2[NPW]
Clear the pre-counter. Trigger to measure and start counting up from 0.
Upload counter to PWTR1[PPW]
EDGE=10
Upload counter to PWTR2[NPW]
EDGE=11
Clear the pre-counter. rigger to measure and start counting up from 0.
Upload counter to PWTR2[NPW]

Figure 27-4. Trigger edge detection and pulse width registers update

PWT_R1[PWTRDY] indicates that the data can be read in PWT_R1[PPW] and/or PWT_R2[NPW], based on the setting of EDGE[1:0].

- When EDGE[1:0] is 00, PWT_R1[PWTRDY] is set whenever PWT_R1[PPW] is updated.
- When EDGE[1:0] is 11, PWT_R1[PWTRDY] is set whenever PWT_R2[NPW] is updated.
- When EDGE[1:0] is 01, PWT_R1[PWTRDY] is set whenever PWTxPPH:L is updated, followed by PWT_R2[NPW]'s update.
- When EDGE[1:0] is 10, PWT_R1[PWTRDY] is set whenever PWT_R2[NPW] is updated, followed by PWT_R1[PPW]'s update.

When PWT_R1[PWTRDY] is set, the updated pulse width register(s) transfers the data to corresponding 16-bit read buffer(s). The read value of pulse width registers actually comes from the corresponding read buffers, whenever the chip is in normal run mode or Debug mode. Reading followed by writing 0 to PWT_R1[PWTRDY] flag clears this field. Until PWT_R1[PWTRDY] is cleared, the 16-bit read buffer(s) cannot be updated. But this does not affect the upload of pulse width registers from the PWT counter.

If another pulse measurement is completed and the pulse width registers are updated, the clearing of the PWT_R1[PWTRDY] flag fails, that is, PWT_R1[PWTRDY] will still be set, but the 16-bit read buffer(s) will be updated again as long as the action is cleared.. The user should complete the pulse width data reading before clearing PWT_R1[PWTRDY] to avoid missing data. This mechanism assures that the second pulse measurement will not be lost in case the MCU does not have enough time to read the first one ready for read. The mechanism is automatically restarted by an MCU reset , writing 1 to PWT_R1[PWTSR] or writing a 0 to PWT_R1[PWTEN] followed by writing a 1 to it.

The following figure illustrates the buffering mechanism of pulse width register:

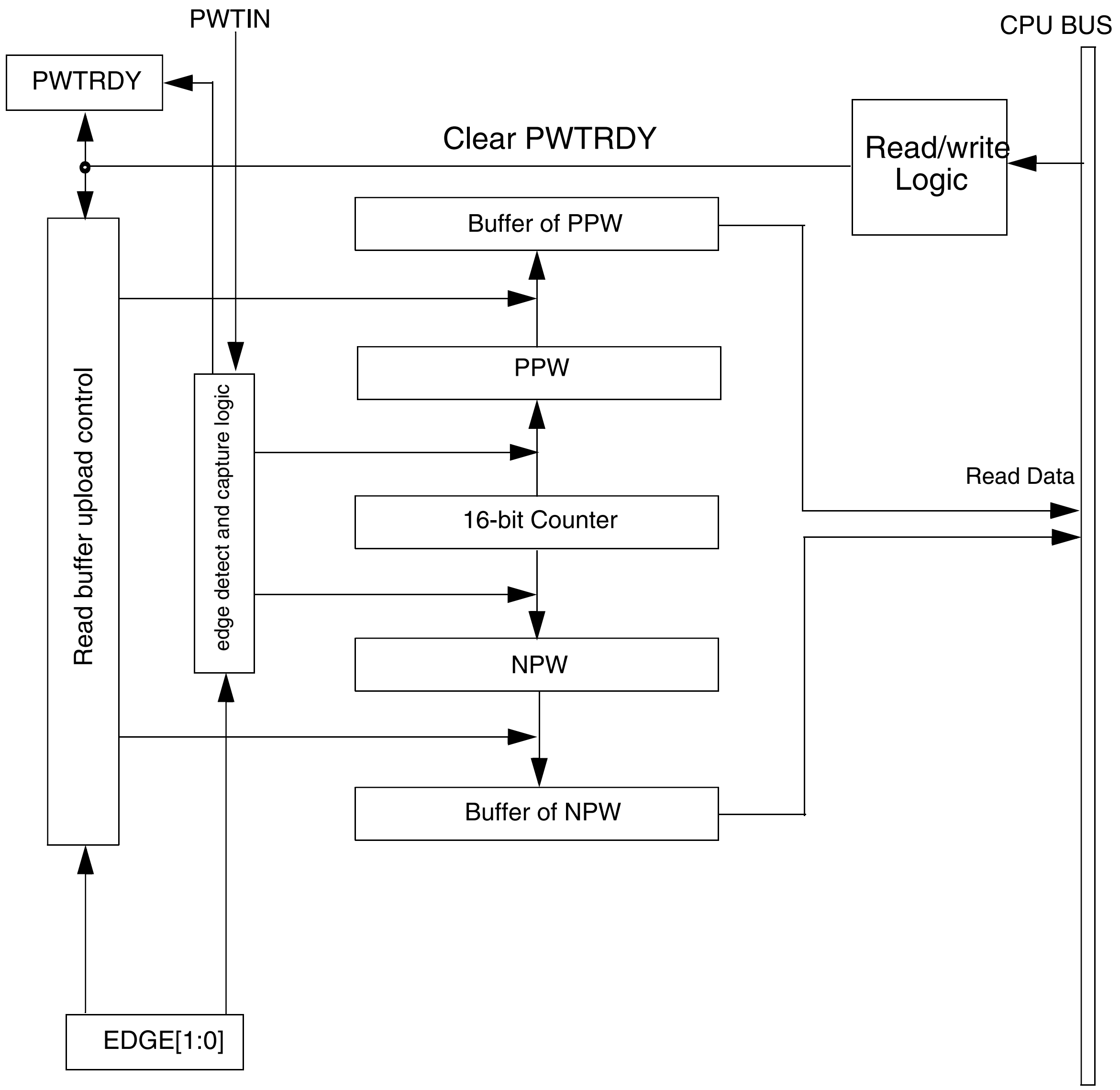


Figure 27-5. Buffering mechanism of Pulse Width register

When PWT completes any pulse width measurement, a signal is generated to reset PWT_R2[CNTC] and the clock prescaler output after the data has been uploaded to the pulse width registers. To assure that there is no missing count, PWT_R2[CNTC] and the clock prescaler output are reset in a bus clock cycle after the completion of a pulse width measurement.

27.5 Reset

27.5.1 General

27.5.2 Description of reset operation

PWT soft reset is built into PWT as a mechanism used to reset/restart the pulse width timer. The PWT soft reset is triggered by writing 1 to PWT_R1[PWTSR]. (This field always reads 0). Unlike reset by the CPU, the PWT reset does not restore everything in the PWT to its reset state. The following steps can be used to describe the reset operation.

1. The PWT counter is set to 0x0000.
2. The 16-bit buffer of PWT counter is reset.
3. The PWT clock prescaler output is reset.
4. The edge detection logic is reset.
5. The capture logic is reset and the latching mechanism of pulse width registers is also restarted.
6. PWT_R1[PPW] and PWT_R2[NPW] are set to 0x0000.
7. PWT_R1[PWTOV] and PWT_R1[PWTRDY] are set to 0.
8. All other PWT register settings are not changed.

Writing a 0 to PWT_R1[PWTEN] also has the above effects except that the reset state will be held until PWT_R1[PWTEN] is set to 1.

27.6 Interrupts

27.6.1 Description of interrupt operation

The other major component of the PWT is the interrupts control logic. When PWT_R1[PWTOV] and PWT_R1[POVIE] are set, a PWT overflow interrupt can be generated. When PWT_R1[PWTRDY] bit and PWT_R1[PRDYIE] are set, a pulse width data ready interrupt can be generated. PWT_R1[PWTIE] controls the interrupt generation of the PWT module. The functionality of the PWT is not affected while the interrupt is being generated.

27.6.2 Application examples

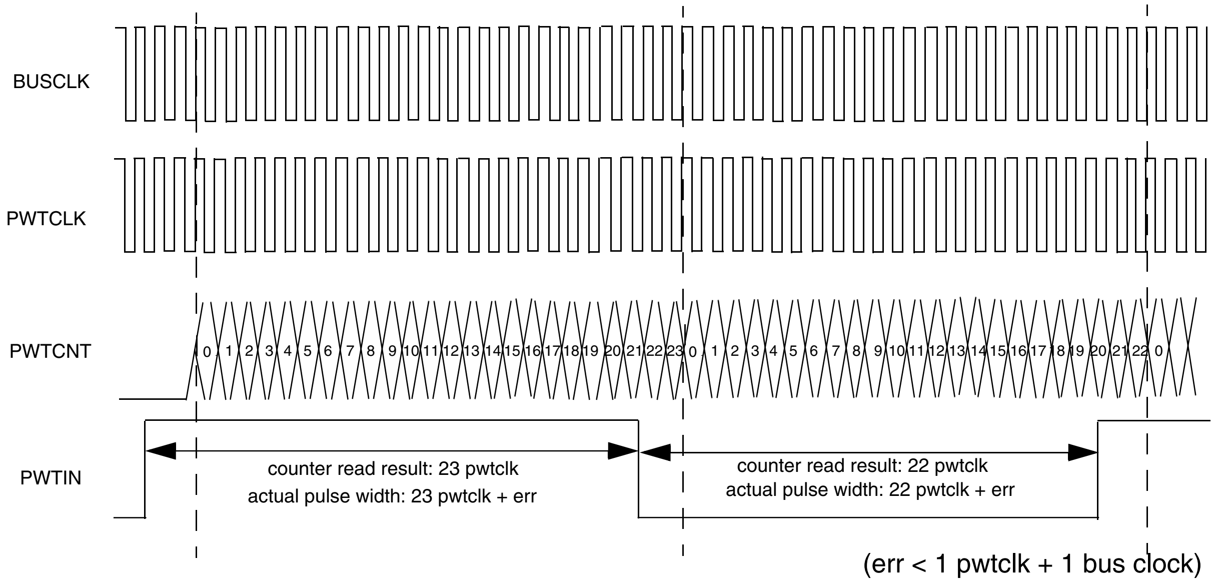

BUSCLK
PWTCLK
PWTCNT
PWTIN
counter read result: 23 pwtclk
actual pulse width: 23 pwtclk + err
counter read result: 22 pwtclk
actual pulse width: 22 pwtclk + err
(err < 1 pwtclk + 1 bus clock)

Figure 27-6. Example at PWTCLK is Bus Clock divided by 1

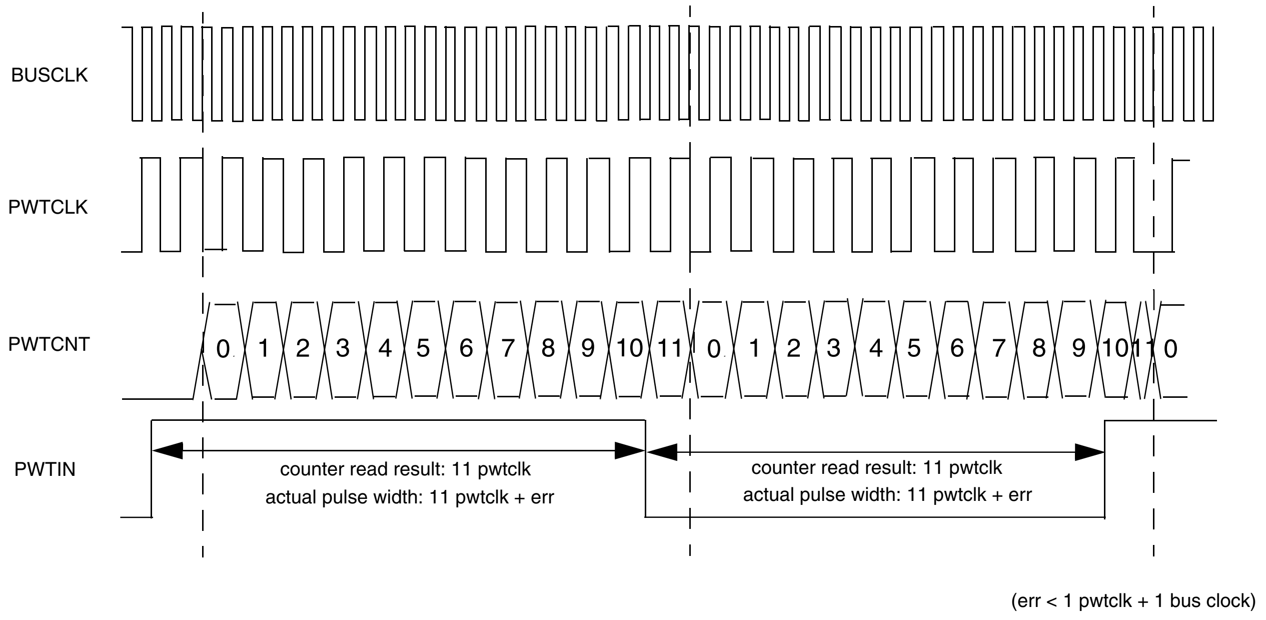

BUSCLK
PWTCLK
PWTCNT
PWTIN
counter read result: 11 pwtclk
actual pulse width: 11 pwtclk + err
counter read result: 11 pwtclk
actual pulse width: 11 pwtclk + err
(err < 1 pwtclk + 1 bus clock)

Figure 27-7. Example at PWTCLK is Bus Clock divided by 2

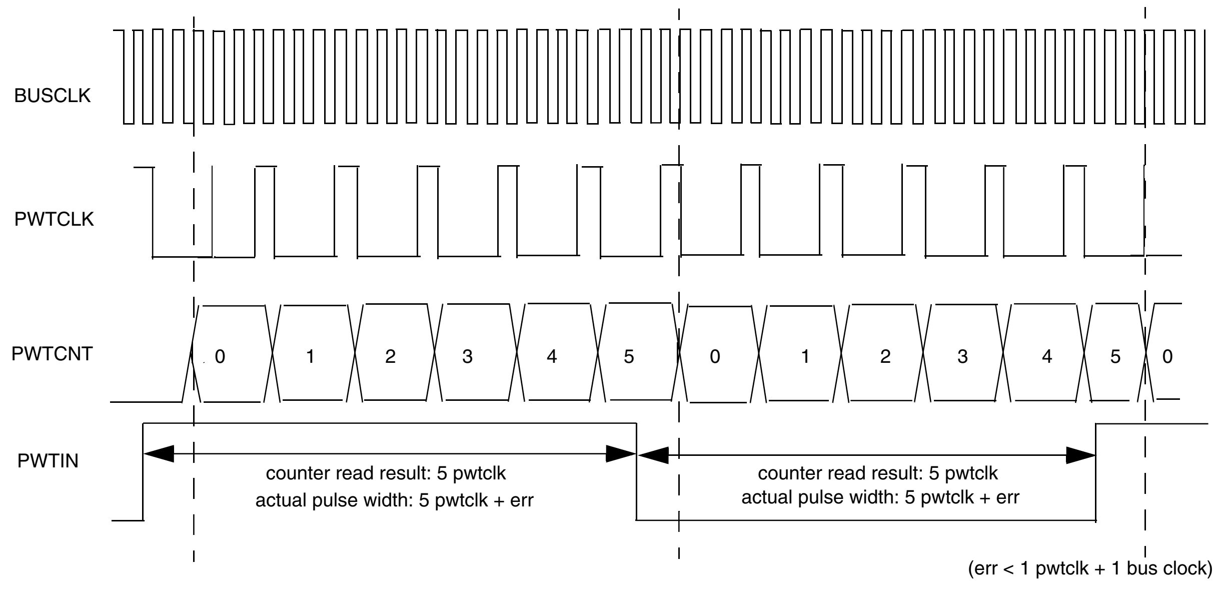

BUSCLK
PWTCLK
PWTCNT
0 1 2 3 4 5 0 1 2 3 4 5 0
PWTIN
counter read result: 5 pwtclk
actual pulse width: 5 pwtclk + err
counter read result: 5 pwtclk
actual pulse width: 5 pwtclk + err
(err < 1 pwtclk + 1 bus clock)

Figure 27-8. Example at PWTCLK is Bus Clock divided by 4

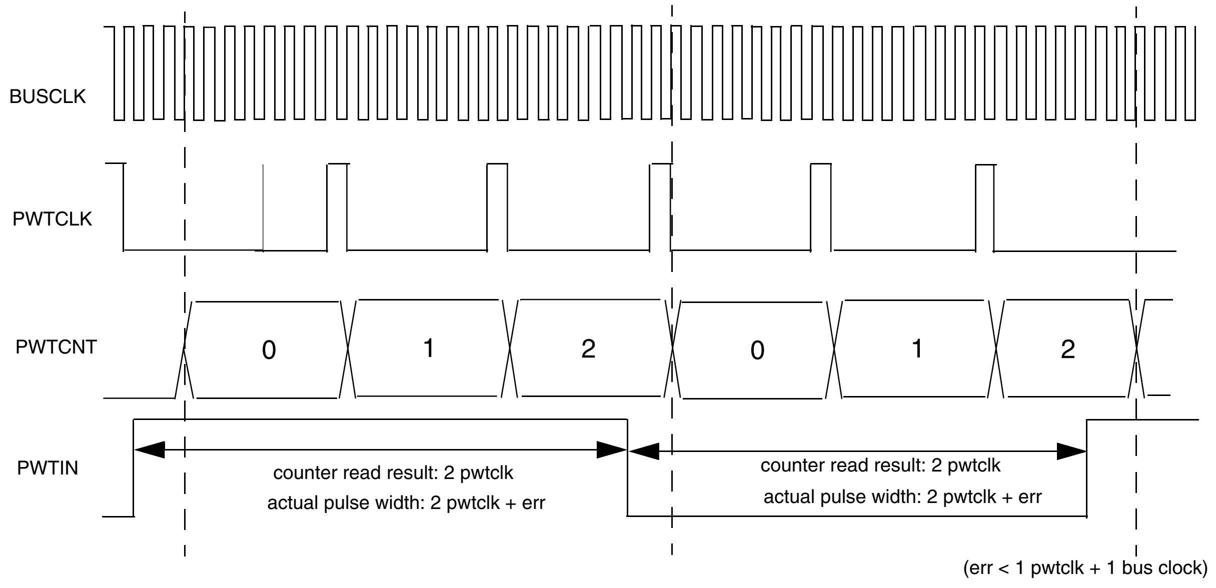

BUSCLK
PWTCLK
PWTCNT
0 1 2 0 1 2
PWTIN
counter read result: 2 pwtclk
actual pulse width: 2 pwtclk + err
counter read result: 2 pwtclk
actual pulse width: 2 pwtclk + err
(err < 1 pwtclk + 1 bus clock)

Figure 27-9. Example at PWTCLK is Bus Clock divided by 8

Chapter 28
Periodic Interrupt Timer (PIT)

28.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The PIT module is an array of timers that can be used to raise interrupts and triggers.

28.1.1 Block diagram

The following figure shows the block diagram of the PIT module.

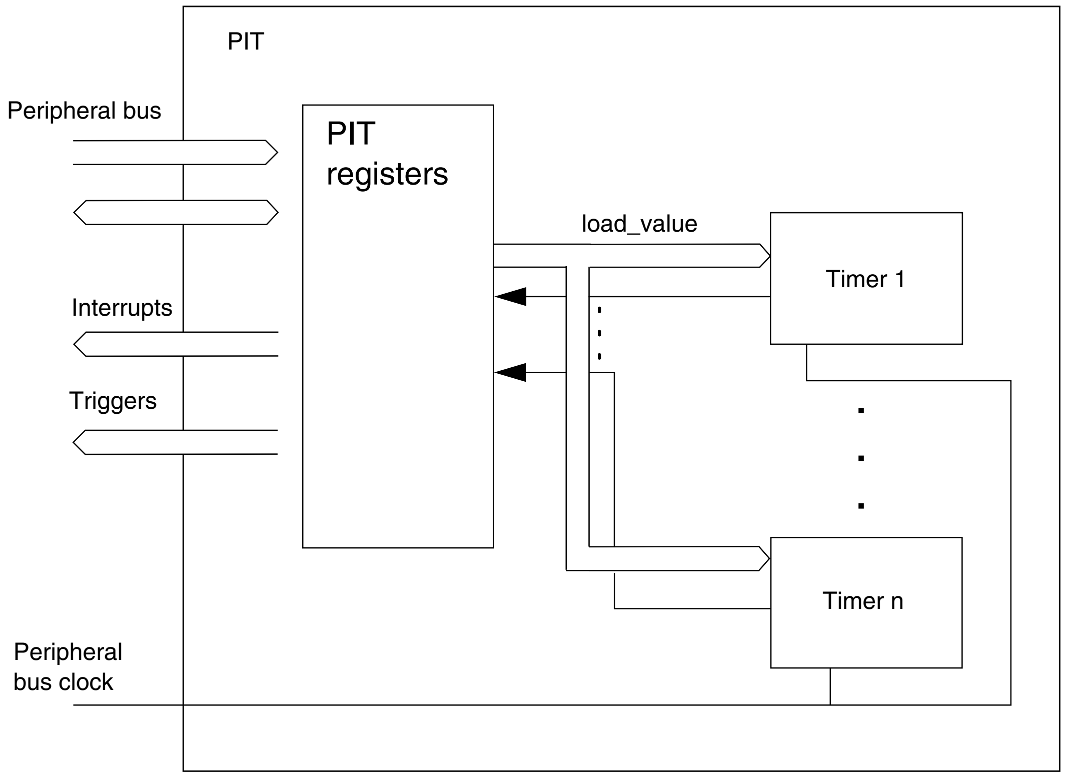


Figure 28-1. Block diagram of the PIT

NOTE

See the chip configuration details for the number of PIT channels used in this MCU.

28.1.2 Features

The main features of this block are:

- Ability of timers to generate trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer

28.2 Signal description

The PIT module has no external pins.

28.3 Memory map/register description

This section provides a detailed description of all registers accessible in the PIT module.

- Reserved registers will read as 0, writes will have no effect.
- See the chip configuration details for the number of PIT channels used in this MCU.

PIT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_7000	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0006h	28.3.1/481
4003_7100	Timer Load Value Register (PIT_LDVAL0)	32	R/W	0000_0000h	28.3.2/482
4003_7104	Current Timer Value Register (PIT_CVAL0)	32	R	0000_0000h	28.3.3/483
4003_7108	Timer Control Register (PIT_TCTRL0)	32	R/W	0000_0000h	28.3.4/483
4003_710C	Timer Flag Register (PIT_TFLG0)	32	R/W	0000_0000h	28.3.5/484
4003_7110	Timer Load Value Register (PIT_LDVAL1)	32	R/W	0000_0000h	28.3.2/482
4003_7114	Current Timer Value Register (PIT_CVAL1)	32	R	0000_0000h	28.3.3/483
4003_7118	Timer Control Register (PIT_TCTRL1)	32	R/W	0000_0000h	28.3.4/483
4003_711C	Timer Flag Register (PIT_TFLG1)	32	R/W	0000_0000h	28.3.5/484

28.3.1 PIT Module Control Register (PIT_MCR)

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

Address: 4003_7000h base + 0h offset = 4003_7000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													Reserved	MDIS	FRZ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

PIT_MCR field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved.
1 MDIS	Module Disable - (PIT section) Disables the standard timers. This field must be enabled before any other setup is done. 0 Clock for standard PIT timers is enabled. 1 Clock for standard PIT timers is disabled.
0 FRZ	Freeze Allows the timers to be stopped when the device enters the Debug mode. 0 Timers continue to run in Debug mode. 1 Timers are stopped in Debug mode.

28.3.2 Timer Load Value Register (PIT_LDVAL*n*)

These registers select the timeout period for the timer interrupts.

Address: 4003_7000h base + 100h offset + (16d × i), where i=0d to 1d

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

PIT_LDVAL*n* field descriptions

Field	Description
TSV	Timer Start Value

PIT_LDVAL*n* field descriptions (continued)

Field	Description
	Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

28.3.3 Current Timer Value Register (PIT_CVAL*n*)

These registers indicate the current timer position.

Address: 4003_7000h base + 104h offset + (16d × i), where i=0d to 1d

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

PIT_CVAL*n* field descriptions

Field	Description
TVL	Current Timer Value Represents the current timer value, if the timer is enabled. **NOTE:** • If the timer is disabled, do not use this field as its value is unreliable. • The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

28.3.4 Timer Control Register (PIT_TCTRL*n*)

These registers contain the control bits for each timer.

Address: 4003_7000h base + 108h offset + (16d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TCTRL*n* field descriptions

Field	Description
31–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 CHN	Chain Mode When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer. For example, for Channel 2, if this field is set, Timer 2 is chained to Timer 1.
1 TIE	Timer Interrupt Enable When an interrupt is pending, or, TFLGn[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TFLGn[TIF] must be cleared first. 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.
0 TEN	Timer Enable Enables or disables the timer. 0 Timer n is disabled. 1 Timer n is enabled.

28.3.5 Timer Flag Register (PIT_TFLG*n*)

These registers hold the PIT interrupt flags.

Address: 4003_7000h base + 10Ch offset + (16d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TFLG*n* field descriptions

Field	Description
31–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PIT_TFLG*n* field descriptions (continued)

Field	Description
0 TIF	Timer Interrupt Flag Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

28.4 Functional description

This section provides the functional description of the module.

28.4.1 General operation

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses and interrupts. Each interrupt is available on a separate interrupt line.

28.4.1.1 Timers

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked by setting TCTRLn[TIE]. A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN]. See the following figure.

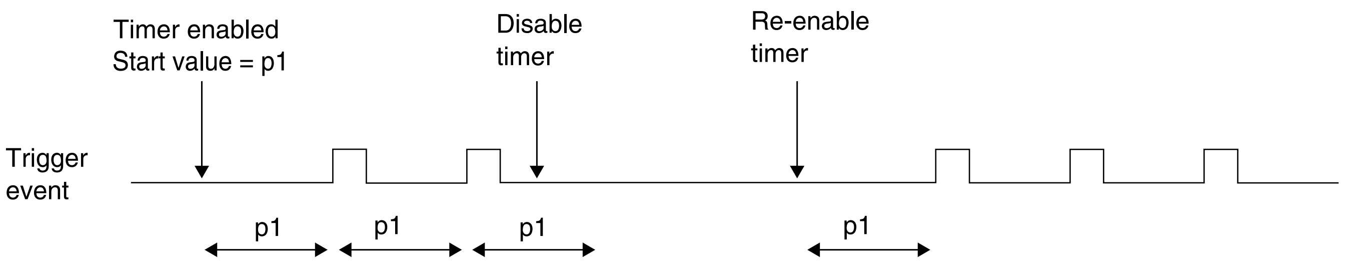


Figure 28-15. Stopping and starting a timer

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value, and then enabling the timer again. See the following figure.

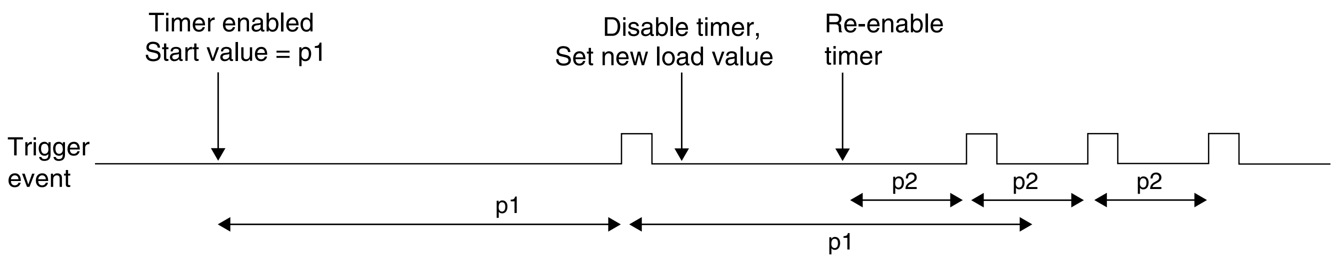


Figure 28-16. Modifying running timer period

It is also possible to change the counter period without restarting the timer by writing LDVAL with the new load value. This value will then be loaded after the next trigger event. See the following figure.

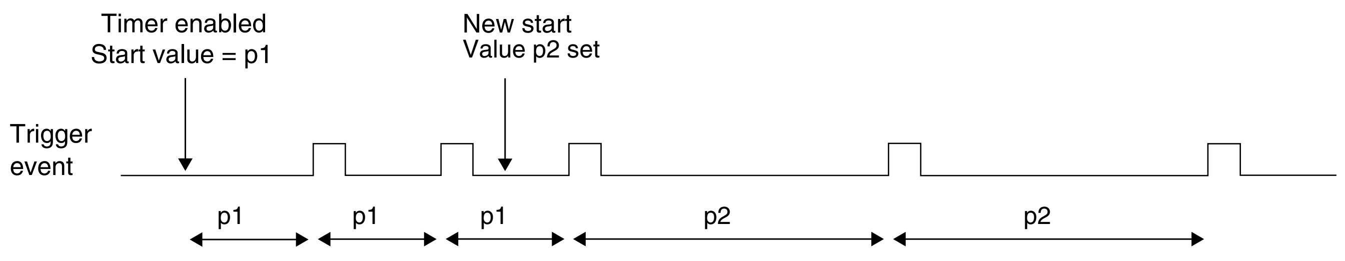


Figure 28-17. Dynamically setting a new load value

28.4.1.2 Debug mode

In Debug mode, the timers will be frozen based on MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system, for example, the timer values, and then continue the operation.

28.4.2 Interrupts

All the timers support interrupt generation. See the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting TCTRLn[TIE]. TFLGn[TIF] are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to the corresponding TFLGn[TIF].

28.4.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

28.5 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

The PIT module must be activated by writing a 0 to MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns. Timer 1 needs to trigger every 5.12 ms/20 ns = 256,000 cycles and Timer 3 every 30 ms/20 ns = 1,500,000 cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) -1

This means LDVAL1 and LDVAL3 must be written with 0x0003E7FF and 0x0016E35F respectively.

The interrupt for Timer 1 is enabled by setting TCTRL1[TIE]. The timer is started by writing 1 to TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore, Timer 3 is started by writing a 1 to TCTRL3[TEN]. TCTRL3[TIE] stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1


// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

28.6 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 hour.

The PIT module needs to be activated by writing a 0 to MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So, Timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to Timer 1 and programmed to trigger 10 times.

The value for the LDVAL register trigger is calculated as number of cycles-1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for Timer 2 is enabled by setting TCTRL2[TIE], the Chain mode is activated by setting TCTRL2[CHN], and the timer is started by writing a 1 to TCTRL2[TEN]. TCTRL1[TEN] needs to be set, and TCTRL1[CHN] and TCTRL1[TIE] are cleared.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 2
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2
```

```
// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```

Chapter 29
Real-Time Counter (RTC)

29.1 Introduction

The real-time counter (RTC) consists of one 16-bit counter, one 16-bit comparator, several binary-based and decimal-based prescaler dividers, three clock sources, one programmable periodic interrupt, and one programmable external toggle pulse output. This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wake-up from low-power modes, Stop and Wait without the need of external components.

29.2 Features

Features of the RTC module include:

- 16-bit up-counter
  - 16-bit modulo match limit
  - Software controllable periodic interrupt on match
- Software selectable clock sources for input to prescaler with programmable 16 bit prescaler
  - OSC 32.768KHz nominal.
  - LPO (~1 kHz)
  - Bus clock
  - Internal reference clock (32 kHz)

29.2.1 Modes of operation

This section defines the RTC operation in Stop, Wait, and Background Debug modes.

29.2.1.1 Wait mode

The RTC continues to run in Wait mode if enabled before executing the WAIT instruction. Therefore, the RTC can be used to bring the MCU out of Wait mode if the real-time interrupt is enabled. For lowest possible current consumption, the RTC must be stopped by software if not needed as an interrupt source during Wait mode.

29.2.1.2 Stop modes

The RTC continues to run in Stop mode if the RTC is enabled before executing the STOP instruction. Therefore, the RTC can be used to bring the MCU out of stop modes with no external components, if the real-time interrupt is enabled.

29.2.2 Block diagram

The block diagram for the RTC module is shown in the following figure.

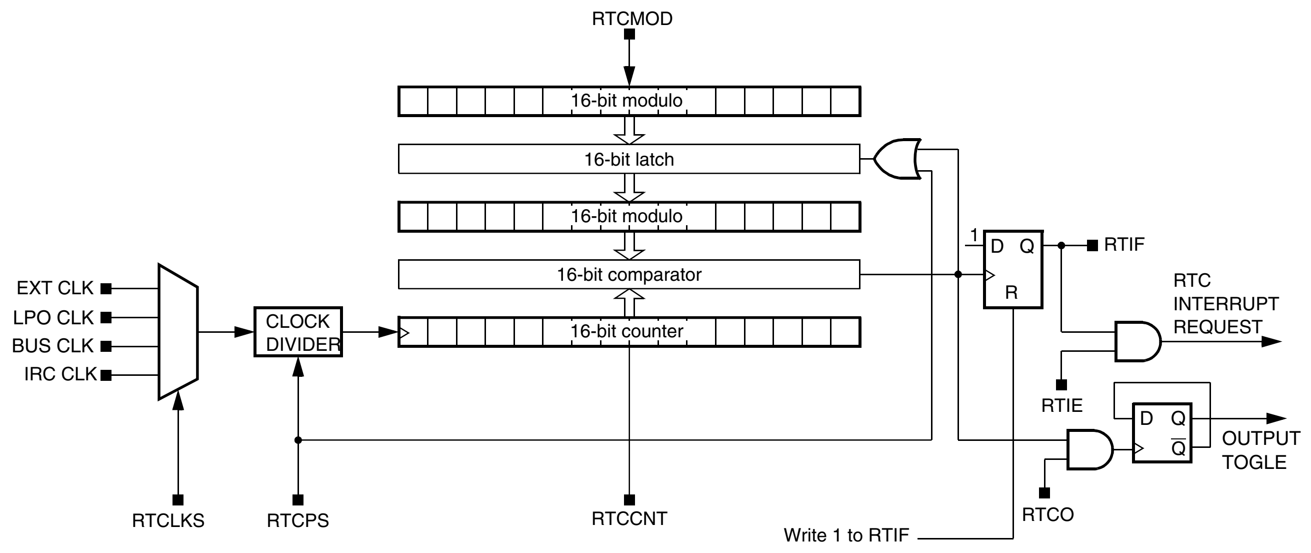


Figure 29-1. Real-time counter (RTC) block diagram

29.3 External signal description

RTCO is the output of RTC. After MCU reset, the RTC_SC[RTCO] is set to high. When the counter overflows, the output is toggled.

29.4 Register definition

The RTC includes a status and control register, a 16-bit counter register, and a 16-bit modulo register.

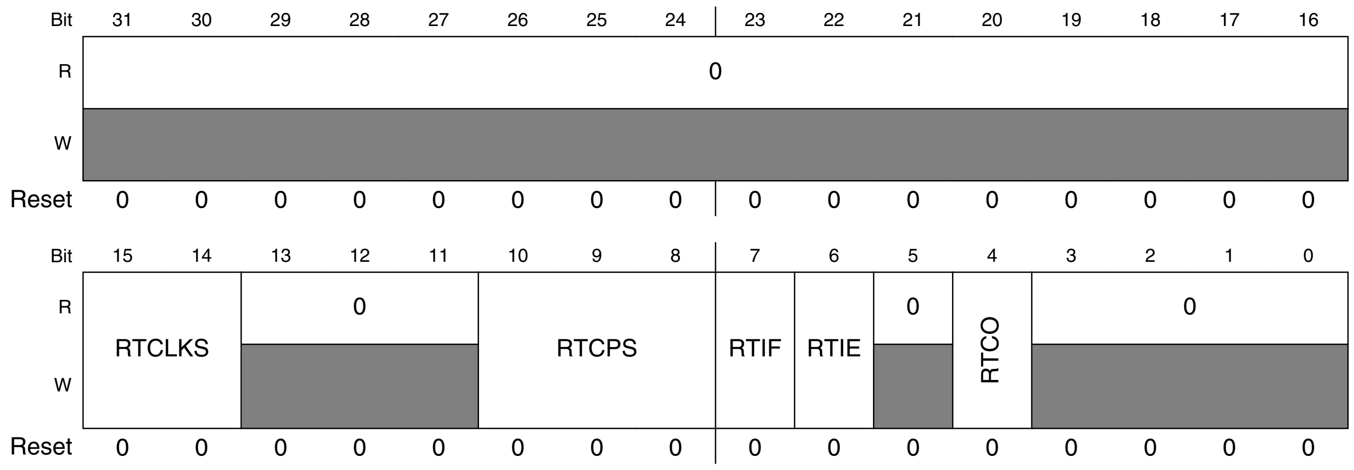
RTC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_D000	RTC Status and Control Register (RTC_SC)	32	R/W	0000_0000h	29.4.1/493
4003_D004	RTC Modulo Register (RTC_MOD)	32	R/W	0000_0000h	29.4.2/495
4003_D008	RTC Counter Register (RTC_CNT)	32	R	0000_0000h	29.4.3/495

29.4.1 RTC Status and Control Register (RTC_SC)

RTC_SC contains the real-time interrupt status flag (RTIF), and the toggle output enable bit (RTCO).

Address: 4003_D000h base + 0h offset = 4003_D000h



RTC_SC field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–14 RTCLKS	Real-Time Clock Source Select This read/write field selects the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters. Reset clears RTCLKS to 00.

Table continues on the next page...

RTC_SC field descriptions (continued)

Field	Description
	00 External clock source. 01 Real-time clock source is 1 kHz (LPOCLK). 10 Internal reference clock (ICSIRCLK). 11 Bus clock.
13–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–8 RTCPS	Real-Time Clock Prescaler Select This read/write field selects binary-based or decimal-based divide-by values for the clock source. Changing the prescaler value clears the prescaler and RTCCNT counters. Reset clears RTCPS to 000. 000 Off 001 If RTCLKS = x0, it is 1; if RTCLKS = x1, it is 128. 010 If RTCLKS = x0, it is 2; if RTCLKS = x1, it is 256. 011 If RTCLKS = x0, it is 4; if RTCLKS = x1, it is 512. 100 If RTCLKS = x0, it is 8; if RTCLKS = x1, it is 1024. 101 If RTCLKS = x0, it is 16; if RTCLKS = x1, it is 2048. 110 If RTCLKS = x0, it is 32; if RTCLKS = x1, it is 100. 111 If RTCLKS = x0, it is 64; if RTCLKS = x1, it is 1000.
7 RTIF	Real-Time Interrupt Flag This status bit indicates the RTC counter register reached the value in the RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RTIF to 0. 0 RTC counter has not reached the value in the RTC modulo register. 1 RTC counter has reached the value in the RTC modulo register.
6 RTIE	Real-Time Interrupt Enable This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set. Reset clears RTIE to 0. 0 Real-time interrupt requests are disabled. Use software polling. 1 Real-time interrupt requests are enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 RTCO	Real-Time Counter Output The read/write bit enables real-time to toggle output on pinout. If this bit is set, the RTCO pinout will be toggled when RTC counter overflows. 0 Real-time counter output disabled. 1 Real-time counter output enabled.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.4.2 RTC Modulo Register (RTC_MOD)

RTC_MOD indicates the value of the 16-bit modulo value.

Address: 4003_D000h base + 4h offset = 4003_D004h

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

RTC_MOD field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
MOD	RTC Modulo This read/write field contains the modulo value used to reset the count to 0x0000 upon a compare match and set SC[RTIF] status field. A value of 0x0000 sets SC[RTIF] on each rising-edge of the prescaler output. Reset sets the modulo to 0x0000.

29.4.3 RTC Counter Register (RTC_CNT)

RTC_CNT indicates the read-only value of the current RTC count of the 16-bit counter.

Address: 4003_D000h base + 8h offset = 4003_D008h

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

RTC_CNT field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
CNT	RTC Count This read-only field contains the current value of the 16-bit counter, read CNT[7:0] first and, then CNT[15:8]. Writes have no effect to this register. Reset or writing different values to SC[RTCLKS] and SC[RTCPS] clear the count to 0x0000.

29.5 Functional description

The RTC is composed of a main 16-bit up-counter with a 16-bit modulo register, a clock source selector, and a prescaler block with binary-based and decimal-based selectable values. The module also contains software selectable interrupt logic and toggle logic for pinout.

After any MCU reset, the counter is stopped and reset to 0x0000, the modulus register is set to 0x0000, and the prescaler is off. The external oscillator clock is selected as the default clock source. To start the prescaler, write any value other than 0 to the Prescaler Select field (RTC_SC[RTCPS]).

The clock sources are software selectable: the external oscillator (OSC), on-chip low power oscillator (LPO), 32-kHz internal reference clock, and bus clock. The RTC Clock Select field (RTC_SC[RTCLKS]) is used to select the desired clock source to the prescaler dividers. If a different value is written to RTC_SC[RTCLKS], the prescaler and CNT counters are reset to 0x00.

RTC_SC[RTCPS] and RTC_SC[RTCLKS] select the desired divide-by value. If a different value is written to RTC_SC[RTCPS], the prescaler and RTCCNT counters are reset to 0x00. The following table shows different prescaler period values.

Table 29-5. Prescaler period

RTCPS	32768Hz OSC clock source prescaler period (RTCLKS = 00)	LPO clock (1 kHz) source prescaler period (RTCLKS = 01)	Internal reference clock (32.768 kHz) source prescaler period (RTCLKS = 10)	Bus clock (8 MHz) source prescaler period (RTCLKS = 11)
000	Off	Off	Off	Off
001	30.5176 µs	128 ms	30.5176 µs	16 µs
010	61.0351 µs	256 ms	61.0351 µs	32 µs
011	122.0703 µs	512 ms	122.0703 µs	64 µs
100	244.1406 µs	1024 ms	244.1406 µs	128 µs
101	488.28125 µs	2048 ms	488.28125 µs	256 µs
110	976.5625 µs	100 ms	976.5625 µs	12.5 µs
111	1.9531 ms	1 s	1.9531 ms	125 µs

The RTC Modulo register (RTC_MOD) allows the compare value to be set to any value from 0x0000 to 0xFFFF. When the counter is active, the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter resets to 0x0000 and continues counting. The Real-Time Interrupt Flag

(RTC_SC[RTIF]) is set whenever a match occurs. The flag sets on the transition from the modulo value to 0x0000. The modulo value written to RTC_MOD is latched until the RTC counter overflows or RTC_SC[RTCPS] is selected nonzero.

The RTC allows for an interrupt to be generated whenever RTC_SC[RTIF] is set. To enable the real-time interrupt, set the Real-Time Interrupt Enable field (RTC_SC[RTIE]). RTC_SC[RTIF] is cleared by writing a 1 to RTC_SC[RTIF].

The RTC also allows an output to external pinout by toggling the level. RTC_SC[RTCO] must be set to enable toggling external pinout. The level depends on the previous state of the pinout when the counter overflows if this function is active.

29.5.1 RTC operation example

This section shows an example of the RTC operation as the counter reaches a matching value from the modulo register.

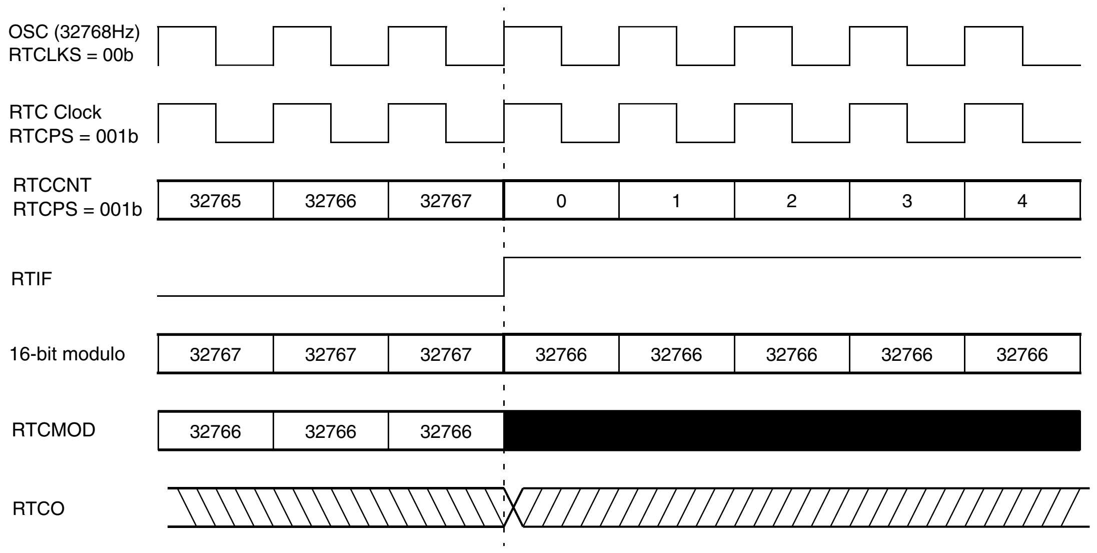


Figure 29-5. RTC counter overflow example

In the above example, the external clock source is selected. The prescaler is set to RTC_SC[RTCPS] = 001b or passthrough. The actual modulo value used by 16-bit comparator is 32767, when the modulo value in the RTC_MOD register is set to 32766. When the counter, RTC_CNT, reaches the modulo value of 32767, the counter overflows to 0x00 and continues counting. The modulo value is updated by fetching from RTC_MOD register. The real-time interrupt flag, RTC_SC[RTIF], sets when the counter value changes from 0x7FFF to 0x0000. The RTC_SC[RTCO] toggles as well when the RTC_SC[RTIF] is set.

29.6 Initialization/application information

This section provides example code to give some basic direction to a user on how to initialize and configure the RTC module. The example software is implemented in C language.

The example below shows how to implement time of day with the RTC using the OSC clock source to achieve the lowest possible power consumption.

Example: 29.6.1 Software calendar implementation in RTC ISR

```
/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;

/* Configure RTC to interrupt every 1 second from OSC (32.768KHz) clock source */
RTC_MOD = 511; // overflow every 32 times
RTC_SC = RTC_SC_RTCPS_MASK; // external 32768 clock selected with 1/64 predivider.
RTC_SC = RTC_SC_RTIF_MASK | RTC_SC_RTIE_MASK; // interrupt cleared and enabled

/**********************************************************************
Function Name : RTC_ISR
Notes : Interrupt service routine for RTC module.
**********************************************************************/
void RTC_ISR(void)
{
/* Clears the interrupt flag, RTIF, and interrupt request */
RTC_SC |= RTC_SC_RTIF_MASK;

/* RTC interrupts every 1 Second */
Seconds++;

/* 60 seconds in a minute */
if (Seconds > 59)
{
Minutes++;
Seconds = 0;
}

/* 60 minutes in an hour */
if (Minutes > 59)
{
Hours++;
Minutes = 0;
}

/* 24 hours in a day */
if (Hours > 23)
{
Days ++;
Hours = 0;
}
}
```

Chapter 30
Serial Peripheral Interface (SPI)

30.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The serial peripheral interface (SPI) module provides for full-duplex, synchronous, serial communication between the MCU and peripheral devices. These peripheral devices can include other microcontrollers, analog-to-digital converters, shift registers, sensors, and memories, among others.

The SPI runs at a baud rate up to the bus clock divided by two in master mode and up to the bus clock divided by four in slave mode. Software can poll the status flags, or SPI operation can be interrupt driven.

NOTE

For the actual maximum SPI baud rate, refer to the Chip Configuration details and to the device’s Data Sheet.

SPI pad cannot be configured as high drive pad.

The SPI also includes a hardware match feature for the receive data buffer.

30.1.1 Features

The SPI includes these distinctive features:

- Master mode or slave mode operation
- Full-duplex or single-wire bidirectional mode

- Programmable transmit bit rate
- Double-buffered transmit and receive data register
- Serial clock phase and polarity options
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Control of SPI operation during wait mode
- Selectable MSB-first or LSB-first shifting
- Receive data buffer hardware match feature

30.1.2 Modes of operation

The SPI functions in the following three modes.

- Run mode

  This is the basic mode of operation.

- Wait mode

  SPI operation in Wait mode is a configurable low power mode, controlled by the SPISWAI bit located in the SPIx_C2 register. In Wait mode, if C2[SPISWAI] is clear, the SPI operates like in Run mode. If C2[SPISWAI] is set, the SPI goes into a power conservative state, with the SPI clock generation turned off. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU enters Run mode. If the SPI is configured as a slave, reception and transmission of a byte continues, so that the slave stays synchronized to the master.

- Stop mode

  To reduce power consumption, the SPI is inactive in stop modes where the peripheral bus clock is stopped but internal logic states are retained. If the SPI is configured as a master, any transmission in progress stops, but is resumed after the CPU enters run mode. If the SPI is configured as a slave, reception and transmission of a data continues, so that the slave stays synchronized to the master.

  The SPI is completely disabled in Stop modes where the peripheral bus clock is stopped and internal logic states are not retained. When the CPU wakes from these Stop modes, all SPI register content is reset.

Detailed descriptions of operating modes appear in Low-power mode options.

30.1.3 Block diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

30.1.3.1 SPI system block diagram

The following figure shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input (SS pin). In this system, the master device has configured its SS pin as an optional slave select output.

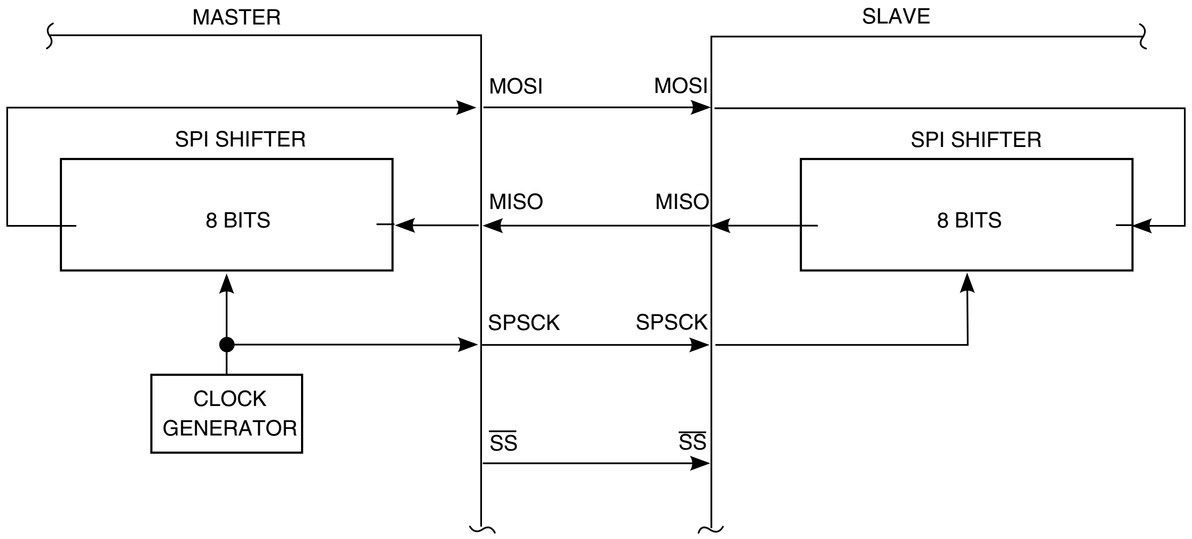


Figure 30-1. SPI system connections

30.1.3.2 SPI module block diagram

The following is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPIx_D) and gets transferred to the SPI Shift Register at the start of a data transfer. After shifting in 8 bits of data, the data is transferred into the double-buffered receiver where it can be read from SPIx_D. Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

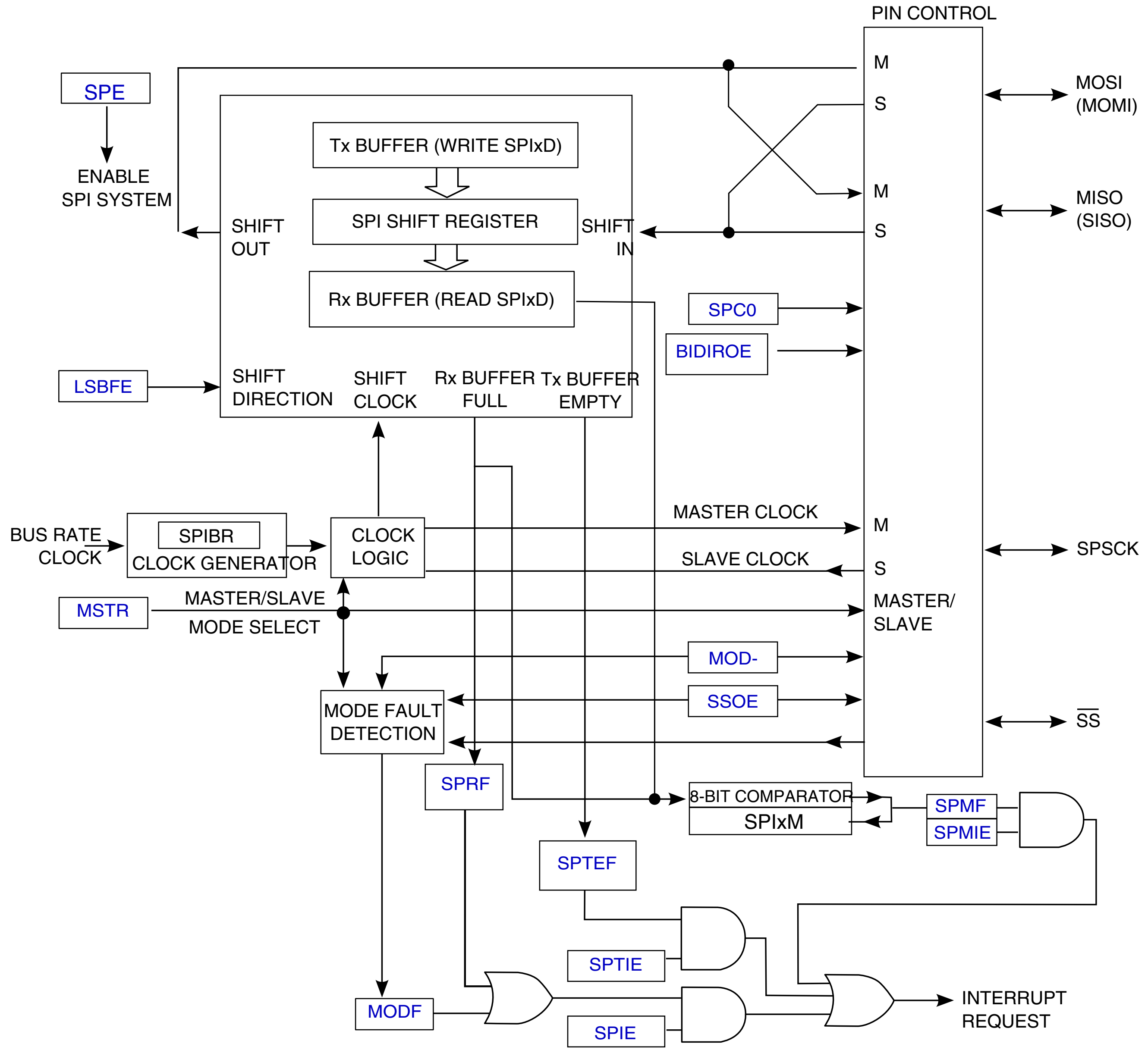


Figure 30-2. SPI module block diagram without FIFO

30.2 External signal description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPE = 0), these four pins revert to other functions that are not controlled by the SPI (based on chip configuration).

30.2.1 SPSCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

30.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 is 0, this pin is the serial data input. If SPC0 is 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE is 0) or an output (BIDIROE is 1). If SPC0 is 1 and slave mode is selected, this pin is not used by the SPI and reverts to other functions (based on chip configuration).

30.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 is 0, this pin is the serial data output. If SPC0 is 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO), and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE is 0) or an output (BIDIROE is 1). If SPC0 is 1 and master mode is selected, this pin is not used by the SPI and reverts to other functions (based on chip configuration).

30.2.4 $\overline{\text{SS}}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN is 0), this pin is not used by the SPI and reverts to other functions (based on chip configuration). When the SPI is enabled as a master and MODFEN is 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE is 0) or as the slave select output (SSOE is 1).

30.3 Memory map/register definition

The SPI has 8-bit registers to select SPI options, to control baud rate, to report SPI status, to hold an SPI data match value, and for transmit/receive data.

SPI memory map

Address offset (hex)	Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	4007_6000	SPI Control Register 1 (SPI0_C1)	8	R/W	04h	30.3.1/505
1	4007_6001	SPI Control Register 2 (SPI0_C2)	8	R/W	00h	30.3.2/507
2	4007_6002	SPI Baud Rate Register (SPI0_BR)	8	R/W	00h	30.3.3/508
3	4007_6003	SPI Status Register (SPI0_S)	8	R	20h	30.3.4/509
5	4007_6005	SPI Data Register (SPI0_D)	8	R/W	00h	30.3.5/510
7	4007_6007	SPI Match Register (SPI0_M)	8	R/W	00h	30.3.6/511

30.3.1 SPI Control Register 1 (SPIx_C1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

Address: 4007_6000h base + 0h offset = 4007_6000h

Bit	7	6	5	4	3	2	1	0
Read Write	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0

SPI0_C1 field descriptions

Field	Description
7 SPIE	SPI Interrupt Enable: for SPRF and MODF Enables the interrupt for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF are inhibited—use polling 1 Request a hardware interrupt when SPRF or MODF is 1
6 SPE	SPI System Enable Enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, the SPI is disabled and forced into an idle state, and all status bits in the S register are reset.

Table continues on the next page...

SPI0_C1 field descriptions (continued)

Field	Description
	0 SPI system inactive 1 SPI system enabled
5 SPTIE	SPI Transmit Interrupt Enable This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). An interrupt occurs when the SPI transmit buffer is empty (SPTEF is set). 0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested
4 MSTR	Master/Slave Mode Select Selects master or slave mode operation. 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	Clock Polarity Selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. This bit effectively places an inverter in series with the clock signal either from a master SPI device or to a slave SPI device. Refer to the description of “SPI Clock Formats” for details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	Clock Phase Selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to the description of “SPI Clock Formats” for details. 0 First edge on SPSCK occurs at the middle of the first cycle of a data transfer. 1 First edge on SPSCK occurs at the start of the first cycle of a data transfer.
1 SSOE	Slave Select Output Enable This bit is used in combination with the Mode Fault Enable (MODFEN) field in the C2 register and the Master/Slave (MSTR) control bit to determine the function of the $\overline{\text{SS}}$ pin. 0 When C2[MODFEN] is 0: In master mode, $\overline{\text{SS}}$ pin function is general-purpose I/O (not SPI). In slave mode, $\overline{\text{SS}}$ pin function is slave select input. When C2[MODFEN] is 1: In master mode, $\overline{\text{SS}}$ pin function is $\overline{\text{SS}}$ input for mode fault. In slave mode, $\overline{\text{SS}}$ pin function is slave select input. 1 When C2[MODFEN] is 0: In master mode, $\overline{\text{SS}}$ pin function is general-purpose I/O (not SPI). In slave mode, $\overline{\text{SS}}$ pin function is slave select input. When C2[MODFEN] is 1: In master mode, $\overline{\text{SS}}$ pin function is automatic $\overline{\text{SS}}$ output. In slave mode: $\overline{\text{SS}}$ pin function is slave select input.
0 LSBFE	LSB First (shifter direction) This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register always have the MSB in bit 7. 0 SPI serial data transfers start with the most significant bit. 1 SPI serial data transfers start with the least significant bit.

30.3.2 SPI Control Register 2 (SPIx_C2)

This read/write register is used to control optional features of the SPI system. Bit 6 is not implemented and always reads 0.

Address: 4007_6000h base + 1h offset = 4007_6001h

Bit	7	6	5	4	3	2	1	0
Read Write	SPMIE	Reserved	Reserved	MODFEN	BIDIROE	Reserved	SPISWAI	SPC0
Reset	0	0	0	0	0	0	0	0

SPI0_C2 field descriptions

Field	Description
7 SPMIE	SPI Match Interrupt Enable This is the interrupt enable bit for the SPI receive data buffer hardware match (SPMF) function. 0 Interrupts from SPMF inhibited (use polling) 1 When SPMF is 1, requests a hardware interrupt
6 Reserved	This field is reserved. Do not write to this reserved bit.
5 Reserved	This field is reserved. Do not write to this reserved bit.
4 MODFEN	Master Mode-Fault Function Enable When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used. For details, refer to the description of the SSOE bit in the C1 register. 0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	Bidirectional Mode Output Enable When bidirectional mode is enabled because SPI pin control 0 (SPC0) is set to 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 is 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
2 Reserved	This field is reserved. Do not write to this reserved bit.
1 SPISWAI	SPI Stop in Wait Mode This bit is used for power conservation while the device is in Wait mode. 0 SPI clocks continue to operate in Wait mode. 1 SPI clocks stop when the MCU enters Wait mode.

Table continues on the next page...

SPI0_C2 field descriptions (continued)

Field	Description
0 SPC0	SPI Pin Control 0 Enables bidirectional pin configurations. 0 SPI uses separate pins for data input and data output (pin mode is normal). In master mode of operation: MISO is master in and MOSI is master out. In slave mode of operation: MISO is slave out and MOSI is slave in. 1 SPI configured for single-wire bidirectional operation (pin mode is bidirectional). In master mode of operation: MISO is not used by SPI; MOSI is master in when BIDIROE is 0 or master I/O when BIDIROE is 1. In slave mode of operation: MISO is slave in when BIDIROE is 0 or slave I/O when BIDIROE is 1; MOSI is not used by SPI.

30.3.3 SPI Baud Rate Register (SPIx_BR)

Use this register to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

Address: 4007_6000h base + 2h offset = 4007_6002h

Bit	7	6	5	4	3	2	1	0
Read	0	SPPR[2:0]			SPR[3:0]			
Write	▨							
Reset	0	0	0	0	0	0	0	0

SPI0_BR field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 SPPR[2:0]	SPI Baud Rate Prescale Divisor This 3-bit field selects one of eight divisors for the SPI baud rate prescaler. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider. Refer to the description of “SPI Baud Rate Generation” for details. 000 Baud rate prescaler divisor is 1. 001 Baud rate prescaler divisor is 2. 010 Baud rate prescaler divisor is 3. 011 Baud rate prescaler divisor is 4. 100 Baud rate prescaler divisor is 5. 101 Baud rate prescaler divisor is 6. 110 Baud rate prescaler divisor is 7. 111 Baud rate prescaler divisor is 8.

Table continues on the next page...

SPI0_BR field descriptions (continued)

Field	Description
SPR[3:0]	SPI Baud Rate Divisor This 4-bit field selects one of nine divisors for the SPI baud rate divider. The input to this divider comes from the SPI baud rate prescaler. Refer to the description of “SPI Baud Rate Generation” for details. 0000 Baud rate divisor is 2. 0001 Baud rate divisor is 4. 0010 Baud rate divisor is 8. 0011 Baud rate divisor is 16. 0100 Baud rate divisor is 32. 0101 Baud rate divisor is 64. 0110 Baud rate divisor is 128. 0111 Baud rate divisor is 256. 1000 Baud rate divisor is 512. All others Reserved

30.3.4 SPI Status Register (SPIx_S)

This register contains read-only status bits. Writes have no meaning or effect.

NOTE

Bits 3 through 0 are not implemented and always read 0.

Address: 4007_6000h base + 3h offset = 4007_6003h

Bit	7	6	5	4	3	2	1	0
Read	SPRF	SPMF	SPTEF	MODF	0			
Write	▨	▨	▨	▨	▨			
Reset	0	0	1	0	0	0	0	0

SPI0_S field descriptions

Field	Description
7 SPRF	SPI Read Buffer Full Flag SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data (D) register. SPRF is cleared by reading SPRF while it is set and then reading the SPI data register. 0 No data available in the receive data buffer 1 Data available in the receive data buffer
6 SPMF	SPI Match Flag SPMF is set after SPRF is 1 when the value in the receive data buffer matches the value in the M register. To clear the flag, read SPMF when it is set and then write a 1 to it. 0 Value in the receive data buffer does not match the value in the M register 1 Value in the receive data buffer matches the value in the M register

Table continues on the next page...

SPI0_S field descriptions (continued)

Field	Description
5 SPTEF	SPI Transmit Buffer Empty Flag This bit is set when the transmit data buffer is empty. SPTEF is cleared by reading the S register with SPTEF set and then writing a data value to the transmit buffer at D. The S register must be read with SPTEF set to 1 before writing data to the D register; otherwise, the D write is ignored. SPTEF is automatically set when all data from the transmit buffer transfers into the transmit shift register. For an idle SPI, data written to D is transferred to the shifter almost immediately so that SPTEF is set within two bus cycles, allowing a second set of data to be queued into the transmit buffer. After completion of the transfer of the data in the shift register, the queued data from the transmit buffer automatically moves to the shifter, and SPTEF is set to indicate that room exists for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter. If a transfer does not stop, the last data that was transmitted is sent out again. 0 SPI transmit buffer not empty 1 SPI transmit buffer empty
4 MODF	Master Mode Fault Flag MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The $\overline{SS}$ pin acts as a mode fault error input only when C1[MSTR] is 1, C2[MODFEN] is 1, and C1[SSOE] is 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1 and then writing to the SPI Control Register 1 (C1). 0 No mode fault error 1 Mode fault error detected
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

30.3.5 SPI Data Register (SPIx_D)

This register is both the input and output register for SPI data. A write to the register writes to the transmit data buffer, allowing data to be queued and transmitted.

When the SPI is configured as a master, data queued in the transmit data buffer is transmitted immediately after the previous transmission has completed.

The SPTEF bit in the S register indicates when the transmit data buffer is ready to accept new data. The S register must be read when S[SPTEF] is set before writing to the SPI data register; otherwise, the write is ignored.

Data may be read from the SPI data register any time after S[SPRF] is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition, and the data from the new transfer is lost. The new data is lost because the receive buffer still held the previous character and was not ready to accept the new data. There is no indication for a receive overrun condition, so the application system designer must ensure that previous data has been read from the receive buffer before a new transfer is initiated.

Address: 4007_6000h base + 5h offset = 4007_6005h

Bit	7	6	5	4	3	2	1	0
Read / Write	Bits[7:0]							
Reset	0	0	0	0	0	0	0	0

SPI0_D field descriptions

Field	Description
Bits[7:0]	Data (low byte)

30.3.6 SPI Match Register (SPIx_M)

This register contains the hardware compare value. When the value received in the SPI receive data buffer equals this hardware compare value, the SPI Match Flag in the S register (S[SPMF]) sets.

Address: 4007_6000h base + 7h offset = 4007_6007h

Bit	7	6	5	4	3	2	1	0
Read / Write	Bits[7:0]							
Reset	0	0	0	0	0	0	0	0

SPI0_M field descriptions

Field	Description
Bits[7:0]	Hardware compare value (low byte)

30.4 Functional description

This section provides the functional description of the module.

30.4.1 General

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI Control Register 1. While C1[SPE] is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select (SS)
- Serial clock (SPSCK)

- Master out/slave in (MOSI)
- Master in/slave out (MISO)

An SPI transfer is initiated in the master SPI device by reading the SPI status register (SPIx_S) when S[SPTEF] = 1 and then writing data to the transmit data buffer (write to SPIxD). When a transfer is complete, received data is moved into the receive data buffer. The SPIxD register acts as the SPI receive data buffer for reads and as the SPI transmit data buffer for writes.

The Clock Phase Control (CPHA) and Clock Polarity Control (CPOL) bits in the SPI Control Register 1 (SPIx_C1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. C1[CPHA] is used to accommodate two fundamentally different protocols by sampling data on odd numbered SPSCK edges or on even numbered SPSCK edges.

The SPI can be configured to operate as a master or as a slave. When the MSTR bit in SPI Control Register 1 is set, master mode is selected; when C1[MSTR] is clear, slave mode is selected.

30.4.2 Master mode

The SPI operates in master mode when C1[MSTR] is set. Only a master SPI module can initiate transmissions. A transmission begins by reading the SPIx_S register while S[SPTEF] = 1 and writing to the master SPI data registers. If the shift register is empty, the byte immediately transfers to the shift register. The data begins shifting out on the MOSI pin under the control of the serial clock.

- SPSCK
  - The SPR3, SPR2, SPR1, and SPR0 baud rate selection bits in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI Baud Rate register control the baud rate generator and determine the speed of the transmission. The SPSCK pin is the SPI clock output. Through the SPSCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.
- MOSI, MISO pin
  - In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.
- $\overline{\text{SS}}$ pin

- If C2[MODFEN] and C1[SSOE] are set, the SS pin is configured as slave select output. The SS output becomes low during each transmission and is high when the SPI is in idle state. If C2[MODFEN] is set and C1[SSOE] is cleared, the $\overline{\text{SS}}$ pin is configured as input for detecting mode fault error. If the SS input becomes low this indicates a mode fault error where another master tries to drive the MOSI and SPSCK lines. In this case, the SPI immediately switches to slave mode by clearing C1[MSTR] and also disables the slave output buffer MISO (or SISO in bidirectional mode). As a result, all outputs are disabled, and SPSCK, MOSI and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state. This mode fault error also sets the Mode Fault (MODF) flag in the SPI Status Register (SPIx_S). If the SPI Interrupt Enable bit (SPIE) is set when S[MODF] gets set, then an SPI interrupt sequence is also requested. When a write to the SPI Data Register in the master occurs, there is a half SPSCK-cycle delay. After the delay, SPSCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI Control Register 1 (see SPI clock formats).

Note

A change of C1[CPOL], C1[CPHA], C1[SSOE], C1[LSBFE], C2[MODFEN], C2[SPC0], C2[BIDIROE] with C2[SPC0] set, SPPR2-SPPR0 and SPR3-SPR0 in master mode abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master has to ensure that the remote slave is set back to idle state.

30.4.3 Slave mode

The SPI operates in slave mode when the MSTR bit in SPI Control Register 1 is clear.

- SPSCK

  In slave mode, SPSCK is the SPI clock input from the master.

- MISO, MOSI pin

  In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit and BIDIROE bit in SPI Control Register 2.

- SS pin

The SS pin is the slave select input. Before a data transmission occurs, the SS pin of the slave SPI must be low. SS must remain low until the transmission is complete. If SS goes high, the SPI is forced into an idle state.

The SS input also controls the serial data output pin. If SS is high (not selected), the serial data output pin is high impedance. If SS is low, the first bit in the SPI Data Register is driven out of the serial data output pin. Also, if the slave is not selected (SS is high), then the SPSCK input is ignored and no internal shifting of the SPI shift register occurs.

Although the SPI is capable of duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin.

Note

When peripherals with duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave's serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI Control Register 1 is clear, odd numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If C1[CPHA] is set, even numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on C1[LSBFE].

When C1[CPHA] is set, the first edge is used to get the first data bit onto the serial data output pin. When C1[CPHA] is clear and the SS input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the eighth shift, the transfer is considered complete and the received data is transferred into the SPI Data register. To indicate transfer is complete, the SPRF flag in the SPI Status Register is set.

Note

A change of the bits C2[BIDIROE] with C2[SPC0] set, C1[CPOL], C1[CPHA], C1[SSOE], C1[LSBFE], C2[MODFEN], and C2[SPC0] in slave mode will corrupt a transmission in progress and must be avoided.

30.4.4 SPI clock formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a Clock Polarity (CPOL) bit and a Clock Phase (CPHA) control bit in the Control Register 1 to select one of four clock formats for data transfers. C1[CPOL] selectively inserts an inverter in series with the clock. C1[CPHA] chooses between two different clock phase relationships between the clock and data.

The following figure shows the clock formats when CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the eighth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in C1[CPOL]. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided C2[MODFEN] and C1[SSOE] = 1). The master $\overline{SS}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

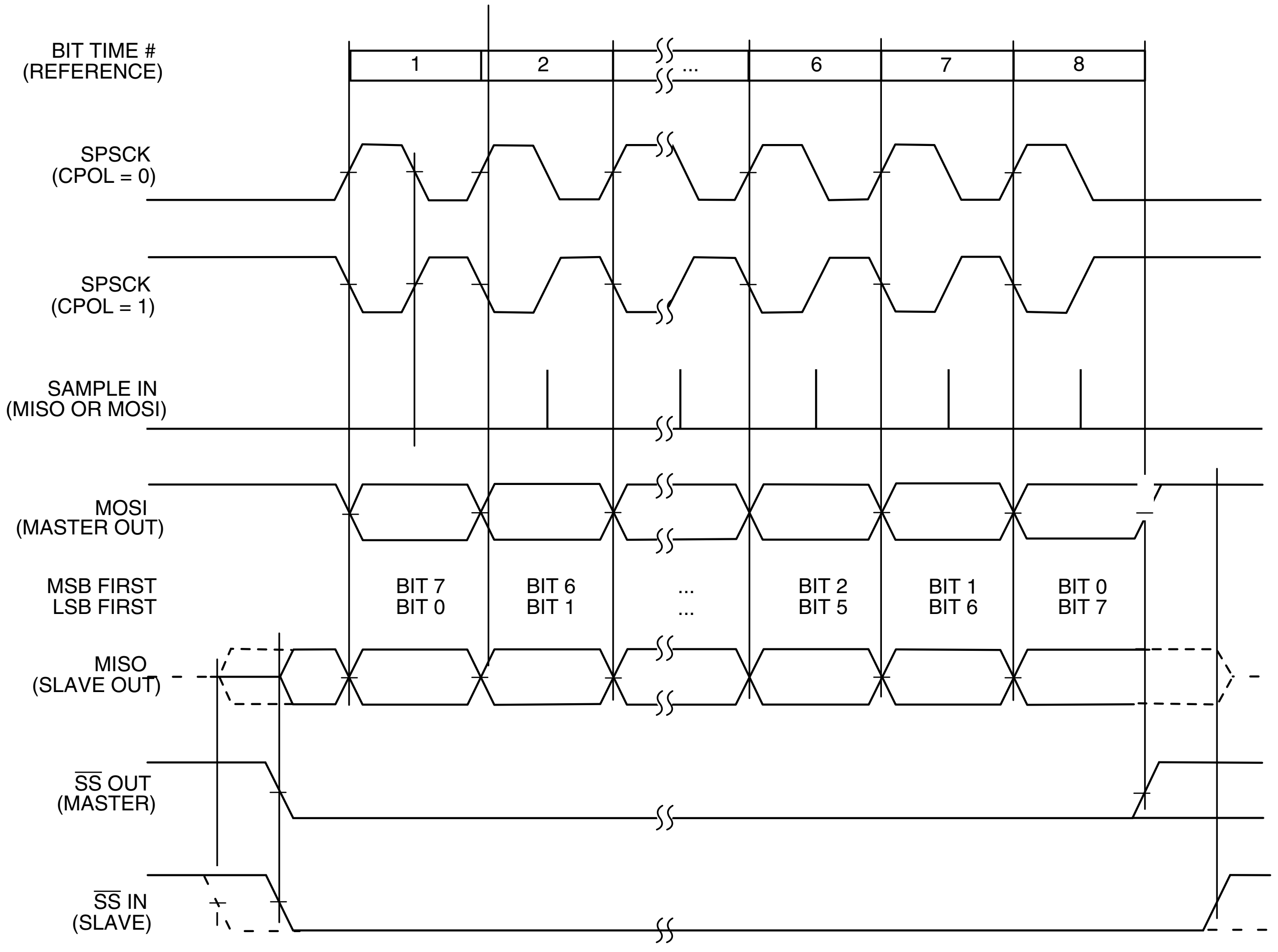


Figure 30-15. SPI clock formats (CPHA = 1)

When C1[CPHA] = 1, the slave begins to drive its MISO output when $\overline{\text{SS}}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively.

When C1[CPHA] = 1, the slave's $\overline{\text{SS}}$ input is not required to go to its inactive high level between transfers. In this clock format, a back-to-back transmission can occur, as follows:

1. A transmission is in progress.
2. A new data byte is written to the transmit buffer before the in-progress transmission is complete.
3. When the in-progress transmission is complete, the new, ready data byte is transmitted immediately.

Between these two successive transmissions, no pause is inserted; the $\overline{\text{SS}}$ pin remains low.

The following figure shows the clock formats when C1[CPHA] = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected (SS IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The SS OUT waveform applies to the slave select output from a master (provided C2[MODFEN] and C1[SSOE] = 1). The master SS output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCK cycle after the end of the eighth bit time of the transfer. The SS IN waveform applies to the slave select input of a slave.

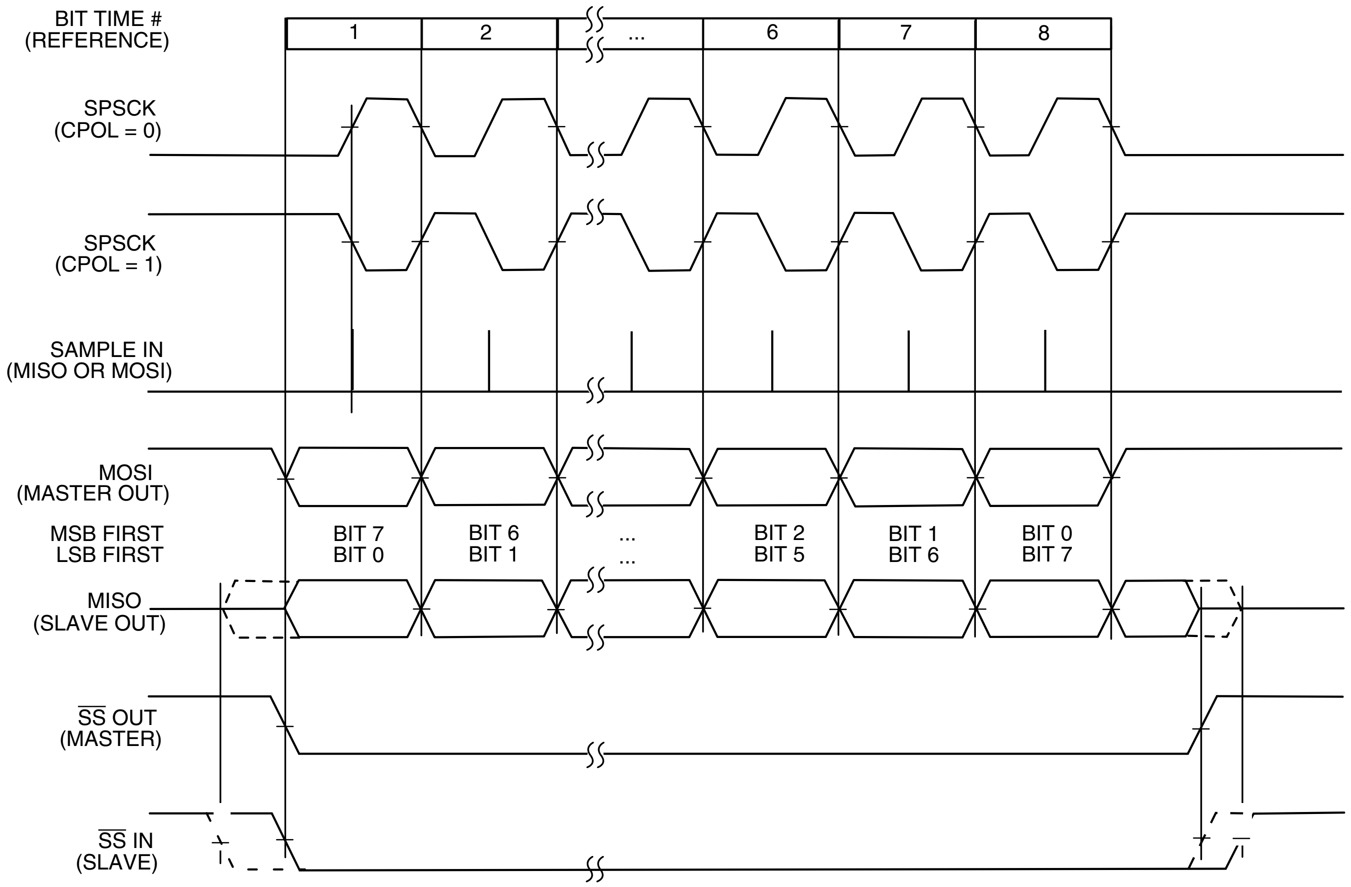


Figure 30-16. SPI clock formats (CPHA = 0)

When C1[CPHA] = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when SS goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When C1[CPHA] = 0, the slave's SS input must go to its inactive high level between transfers.

30.4.5 SPI baud rate generation

As shown in the following figure, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR3:SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, 256, or 512 to get the internal SPI master mode bit-rate clock.

The baud rate generator is activated only when the SPI is in the master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease I_{DD} current.

The baud rate divisor equation is as follows (except those reserved combinations in the SPI Baud Rate Divisor table).

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \times 2^{(\text{SPR} + 1)}$$

The baud rate can be calculated with the following equation:

```
BaudRate = BusClock / BaudRateDivisor
```

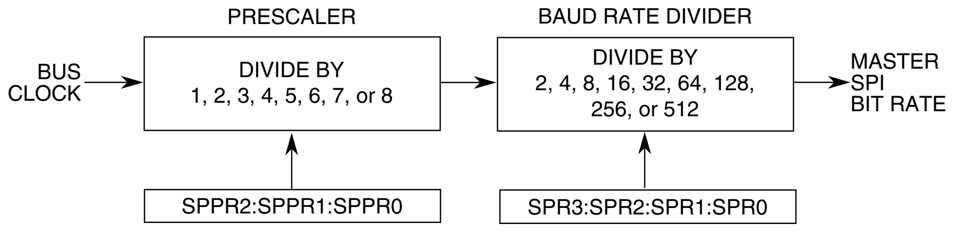


Figure 30-17. SPI baud rate generation

30.4.6 Special features

The following section describes the special features of SPI module.

30.4.6.1 $\overline{SS}$ Output

The $\overline{SS}$ output feature automatically drives the $\overline{SS}$ pin low during transmission to select external devices and drives the $\overline{SS}$ pin high during idle to deselect external devices. When the $\overline{SS}$ output is selected, the $\overline{SS}$ output pin is connected to the $\overline{SS}$ input pin of the external device.

The $\overline{SS}$ output is available only in master mode during normal SPI operation by asserting C1[SSOE] and C2[MODFEN] as shown in the description of C1[SSOE].

The mode fault feature is disabled while $\overline{SS}$ output is enabled.

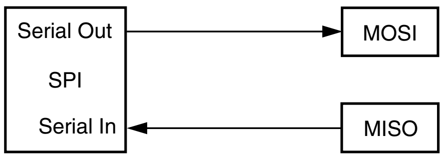
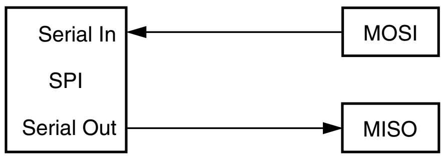
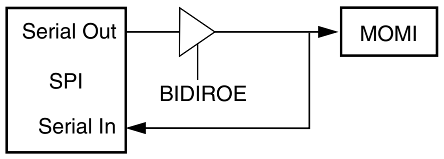
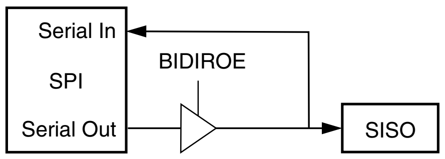
Note

Be careful when using the $\overline{SS}$ output feature in a multimaster system because the mode fault feature is not available for detecting system errors between masters.

30.4.6.2 Bidirectional mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI Control Register 2 (see the following table). In this mode, the SPI uses only one serial data pin for the interface with one or more external devices. C1[MSTR] decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The MISO pin in master mode and MOSI pin in slave mode are not used by the SPI.

Table 30-15. Normal Mode and Bidirectional Mode

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
Normal Mode **SPC0 = 0**		
Bidirectional Mode **SPC0 = 1**		

The direction of each serial I/O pin depends on C2[BIDIROE]. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

The SPSCK is an output for the master mode and an input for the slave mode.

$\overline{\text{SS}}$ is the input or output for the master mode, and it is always the input for the slave mode.

The bidirectional mode does not affect SPSCK and $\overline{\text{SS}}$ functions.

Note

In bidirectional master mode, with the mode fault feature enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode. In this case, MISO becomes occupied by the SPI and MOSI is not used. Consider this scenario if the MISO pin is used for another purpose.

30.4.7 Error conditions

The SPI module has one error condition: the mode fault error.

30.4.7.1 Mode fault error

If the $\overline{\text{SS}}$ input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SPSCK lines simultaneously. This condition is not permitted in normal operation, and it sets the MODF bit in the SPI status register automatically provided that C2[MODFEN] is set.

In the special case where the SPI is in master mode and C2[MODFEN] is cleared, the $\overline{\text{SS}}$ pin is not used by the SPI. In this special case, the mode fault error function is inhibited and MODF remains cleared. If the SPI system is configured as a slave, the $\overline{\text{SS}}$ pin is a dedicated input pin. A mode fault error does not occur in slave mode.

If a mode fault error occurs, the SPI is switched to slave mode, with the exception that the slave output buffer is disabled. So the SPSCK, MISO and MOSI pins are forced to be high impedance inputs to avoid any possibility of conflict with another output driver. A transmission in progress is aborted and the SPI is forced into idle state.

If the mode fault error occurs in the bidirectional mode for an SPI system configured in master mode, the output enable of MOMI (MOSI in bidirectional mode) is cleared if it was set. No mode fault error occurs in the bidirectional mode for the SPI system configured in slave mode.

The mode fault flag is cleared automatically by a read of the SPI Status Register (with MODF set) followed by a write to SPI Control Register 1. If the mode fault flag is cleared, the SPI becomes a normal master or slave again.

30.4.8 Low-power mode options

This section describes the low-power mode options.

30.4.8.1 SPI in Run mode

In Run mode, with the SPI system enable (SPE) bit in the SPI Control Register 1 clear, the SPI system is in a low-power, disabled state. SPI registers can still be accessed, but clocks to the core of this module are disabled.

30.4.8.2 SPI in Wait mode

SPI operation in Wait mode depends upon the state of the SPISWAI bit in SPI Control Register 2.

- If C2[SPISWAI] is clear, the SPI operates normally when the CPU is in Wait mode.
- If C2[SPISWAI] is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
    - If C2[SPISWAI] is set and the SPI is configured for master, any transmission and reception in progress stops at Wait mode entry. The transmission and reception resumes when the SPI exits Wait mode.
    - If C2[SPISWAI] is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SPSCK continues to be driven from the master. This keeps the slave synchronized to the master and the SPSCK.

      If the master transmits data while the slave is in wait mode, the slave continues to send data consistent with the operation mode at the start of wait mode (that is, if the slave is currently sending its SPIx_D to the master, it continues to send the same byte. Otherwise, if the slave is currently sending the last data received byte from the master, it continues to send each previously received data from the master byte).

Note

Care must be taken when expecting data from a master while the slave is in a Wait mode or a Stop mode where the peripheral bus clock is stopped but internal logic states are retained. Even though the shift register continues to operate, the rest of the SPI is shut down (that is, an SPRF interrupt is not generated until an exit from Stop or Wait mode). Also, the data from the shift register is not copied into the SPIx_D registers until after the slave SPI has exited Wait or Stop mode. An SPRF flag and SPIx_D copy is only generated if Wait mode is entered or exited during a transmission. If the slave enters Wait mode in idle mode and exits Wait mode in idle mode, neither an SPRF nor a SPIx_D copy occurs.

30.4.8.3 SPI in Stop mode

Operation in a Stop mode where the peripheral bus clock is stopped but internal logic states are retained depends on the SPI system. The Stop mode does not depend on C2[SPISWAI]. Upon entry to this type of stop mode, the SPI module clock is disabled (held high or low).

- If the SPI is in master mode and exchanging data when the CPU enters the Stop mode, the transmission is frozen until the CPU exits stop mode. After the exit from stop mode, data to and from the external SPI is exchanged correctly.
- In slave mode, the SPI remains synchronized with the master.

The SPI is completely disabled in a stop mode where the peripheral bus clock is stopped and internal logic states are not retained. After an exit from this type of stop mode, all registers are reset to their default values, and the SPI module must be reinitialized.

30.4.9 Reset

The reset values of registers and signals are described in the Memory Map and Register Descriptions content, which details the registers and their bitfields.

- If a data transmission occurs in slave mode after a reset without a write to SPIx_D, the transmission consists of "garbage" or the data last received from the master before the reset.
- Reading from SPIx_D after reset always returns zeros.

30.4.10 Interrupts

The SPI originates interrupt requests only when the SPI is enabled (the SPE bit in the SPIx_C1 register is set). The following is a description of how the SPI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt priority are chip dependent.

Four flag bits, three interrupt mask bits, and one interrupt vector are associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). The SPI match interrupt enable mask bit (SPIMIE) enables interrupts from the SPI match flag (SPMF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check the flag bits to determine which event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

30.4.10.1 MODF

MODF occurs when the master detects an error on the $\overline{SS}$ pin. The master SPI must be configured for the MODF feature (see the description of the C1[SSOE] bit). Once MODF is set, the current transfer is aborted and the master (MSTR) bit in the SPIx_C1 register resets to 0.

The MODF interrupt is reflected in the status register's MODF flag. Clearing the flag also clears the interrupt. This interrupt stays active while the MODF flag is set. MODF has an automatic clearing process that is described in the SPI Status Register.

30.4.10.2 SPRF

SPRF occurs when new data has been received and copied to the SPI receive data buffer.

After SPRF is set, it does not clear until it is serviced. SPRF has an automatic clearing process that is described in the SPI Status Register details. If the SPRF is not serviced before the end of the next transfer (that is, SPRF remains active throughout another transfer), the subsequent transfers are ignored and no new data is copied into the Data register.

30.4.10.3 SPTEF

SPTEF occurs when the SPI transmit buffer is ready to accept new data.

After SPTEF is set, it does not clear until it is serviced. SPTEF has an automatic clearing process that is described in the SPI Status Register details.

30.4.10.4 SPMF

SPMF occurs when the data in the receive data buffer is equal to the data in the SPI Match Register.

30.4.10.5 Asynchronous interrupt in low-power modes

When the CPU is in Wait mode or Stop mode and the SPI module receives a transmission, the SPI module can generate an asynchronous interrupt to wake the CPU from the low power mode. The module generates the asynchronous interrupt only when all of the following conditions apply:

1. C1[SPIE] is set to 1.
2. The CPU is in Wait mode—in which case C2[SPISWAI] must be 1—or in Stop mode where the peripheral bus clock is stopped but internal logic states are retained.
3. The SPI module is in slave mode.
4. The received transmission ends.

After the interrupt wakes the CPU and the peripheral bus clock is active again, the SPI module copies the received data from the shifter into the Data register and generates flags signals. During the wakeup phase, a continuous transmission from a master would destroy the first received data.

30.5 Initialization/application information

This section discusses an example of how to initialize and use the SPI.

30.5.1 Initialization sequence

Before the SPI module can be used for communication, an initialization procedure must be carried out, as follows:

1. Update the Control Register 1 (SPIx_C1) to enable the SPI and to control interrupt enables. This register also sets the SPI as master or slave, determines clock phase and polarity, and configures the main SPI options.
2. Update the Control Register 2 (SPIx_C2) to enable additional SPI functions such as the SPI match interrupt feature, the master mode-fault function, and bidirectional mode output as well as to control and other optional features.
3. Update the Baud Rate Register (SPIx_BR) to set the prescaler and bit rate divisor for an SPI master.
4. Update the Hardware Match Register (SPIx_M) with the value to be compared to the receive data register for triggering an interrupt if hardware match interrupts are enabled.
5. In the master, read SPIx_S while S[SPTEF] = 1, and then write to the transmit data register (SPIx_D) to begin transfer.

30.5.2 Pseudo-Code Example

In this example, the SPI module is set up for master mode with only hardware match interrupts enabled. The SPI runs at a maximum baud rate of bus clock divided by 2. Clock phase and polarity are set for an active-high SPI clock where the first edge on SPSCK occurs at the start of the first cycle of a data transfer.

SPIx_C1=0x54(%01010100)				
Bit 7	SPIE	=	0	Disables receive and mode fault interrupts
Bit 6	SPE	=	1	Enables the SPI system
Bit 5	SPTIE	=	0	Disables SPI transmit interrupts
Bit 4	MSTR	=	1	Sets the SPI module as a master SPI device
Bit 3	CPOL	=	0	Configures SPI clock as active-high
Bit 2	CPHA	=	1	First edge on SPSCK at start of first data transfer cycle
Bit 1	SSOE	=	0	Determines SS pin function when mode fault enabled
Bit 0	LSBFE	=	0	SPI serial data transfers start with most significant bit

SPIx_C2 = 0x80(%10000000)				
Bit 7	SPMIE	=	1	SPI hardware match interrupt enabled
Bit 6		=	0	Unimplemented
Bit 5		=	0	Reserved
Bit 4	MODFEN	=	0	Disables mode fault function
Bit 3	BIDIROE	=	0	SPI data I/O pin acts as input

Table continues on the next page...

SPIx_C2 = 0x80(%10000000)				
Bit 2		=	0	Reserved
Bit 1	SPISWAI	=	0	SPI clocks operate in wait mode
Bit 0	SPC0	=	0	uses separate pins for data input and output

SPIx_BR = 0x00(%00000000)				
Bit 7		=	0	Reserved
Bit 6:4		=	000	Sets prescale divisor to 1
Bit 3:0		=	0000	Sets baud rate divisor to 2

SPIx_S = 0x00(%00000000)				
Bit 7	SPRF	=	0	Flag is set when receive data buffer is full
Bit 6	SPMF	=	0	Flag is set when SPIx_M = receive data buffer
Bit 5	SPTEF	=	0	Flag is set when transmit data buffer is empty
Bit 4	MODF	=	0	Mode fault flag for master mode
Bit 3:0		=	0	Reserved

SPIx_M = 0xXX
Holds bits 0–7 of the hardware match buffer.

SPIx_D = 0xxx
Holds bits 0–7 of the data to be transmitted by the transmit buffer and received by the receive buffer.

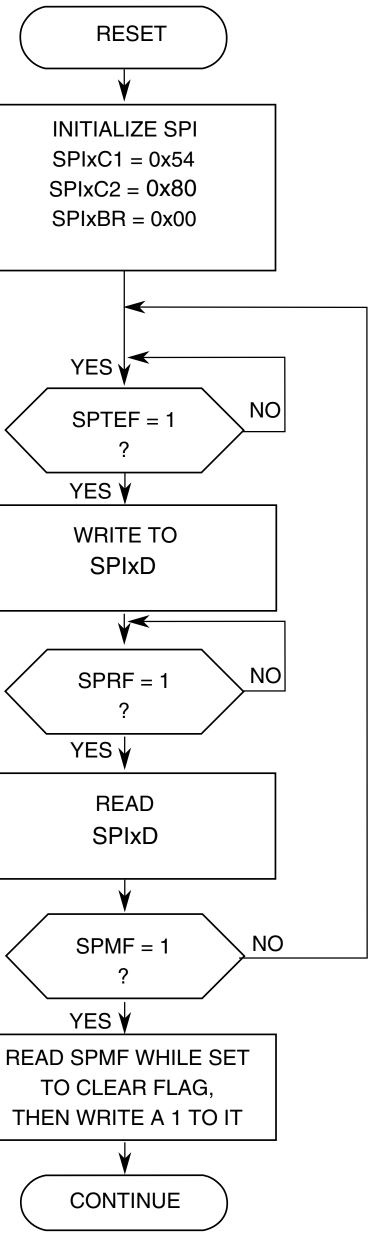


Figure 30-18. Initialization Flowchart Example for SPI Master Device

Chapter 31
Inter-Integrated Circuit (I2C)

31.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The inter-integrated circuit (I^2C, I2C, or IIC) module provides a method of communication between a number of devices.

The interface is designed to operate up to 100 kbit/s with maximum bus loading and timing. The I2C device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF. The I2C module also complies with the *System Management Bus (SMBus) Specification, version 2*.

31.1.1 Features

The I2C module has the following features:

- Compatible with *The I^2C-Bus Specification*
- Multimaster operation
- Software programmable for one of 64 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation and detection
- Repeated START signal generation and detection
- Acknowledge bit generation and detection

- Bus busy detection
- General call recognition
- 10-bit address extension
- Support for *System Management Bus (SMBus) Specification, version 2*
- Programmable glitch input filter
- Low power mode wakeup on slave address match
- Range slave address support

31.1.2 Modes of operation

The I2C module's operation in various low power modes is as follows:

- Run mode: This is the basic mode of operation. To conserve power in this mode, disable the module.
- Wait mode: The module continues to operate when the core is in Wait mode and can provide a wakeup interrupt.
- Stop mode: The module is inactive in Stop mode for reduced power consumption, except that address matching is enabled in Stop mode. The STOP instruction does not affect the I2C module's register states.

31.1.3 Block diagram

The following figure is a functional block diagram of the I2C module.

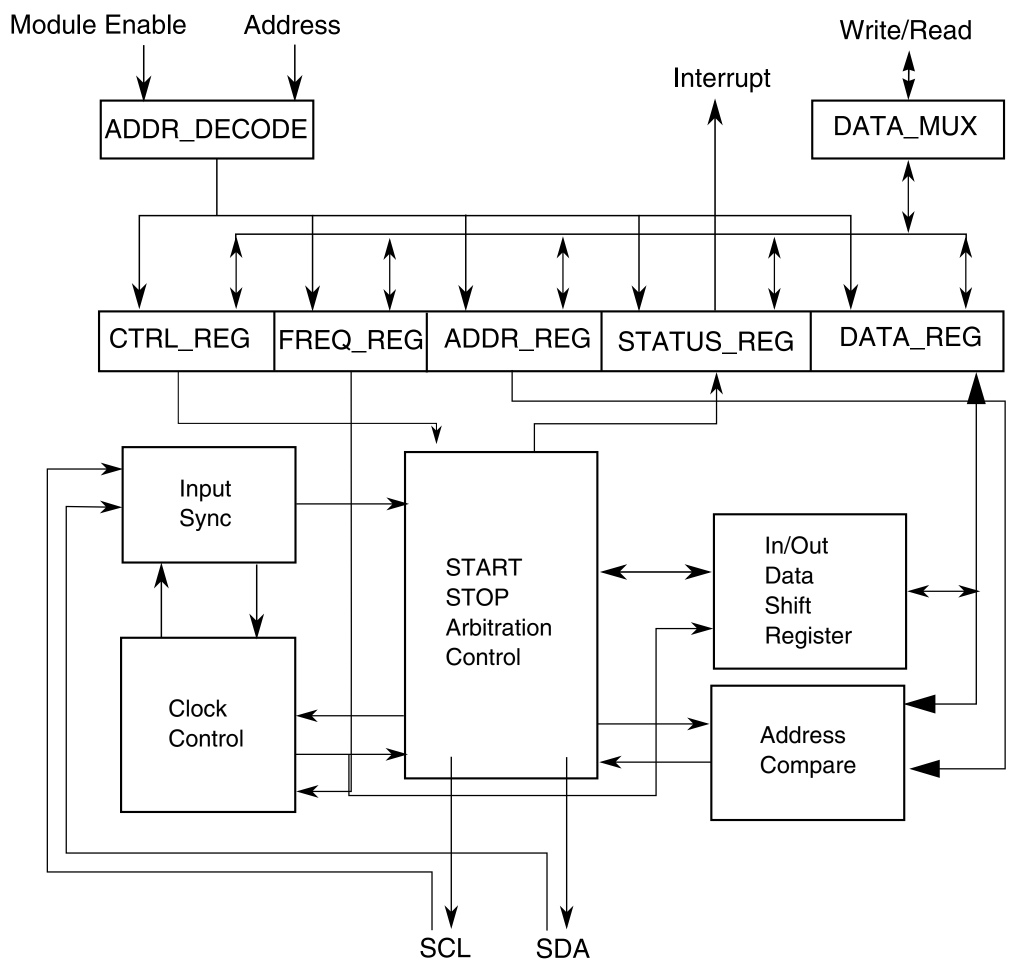


Figure 31-1. I2C Functional block diagram

31.2 I^{2}C signal descriptions

The signal properties of I^{2}C are shown in the table found here.

Table 31-1. I^{2}C signal descriptions

Signal	Description	I/O
SCL	Bidirectional serial clock line of the I^{2}C system.	I/O
SDA	Bidirectional serial data line of the I^{2}C system.	I/O

31.3 Memory map/register definition

This section describes in detail all I2C registers accessible to the end user.

I2C memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_6000	I2C Address Register 1 (I2C0_A1)	8	R/W	00h	31.3.1/532
4006_6001	I2C Frequency Divider register (I2C0_F)	8	R/W	00h	31.3.2/533
4006_6002	I2C Control Register 1 (I2C0_C1)	8	R/W	00h	31.3.3/534
4006_6003	I2C Status register (I2C0_S)	8	R/W	80h	31.3.4/535
4006_6004	I2C Data I/O register (I2C0_D)	8	R/W	00h	31.3.5/537
4006_6005	I2C Control Register 2 (I2C0_C2)	8	R/W	00h	31.3.6/538
4006_6006	I2C Programmable Input Glitch Filter register (I2C0_FLT)	8	R/W	00h	31.3.7/539
4006_6007	I2C Range Address register (I2C0_RA)	8	R/W	00h	31.3.8/540
4006_6008	I2C SMBus Control and Status register (I2C0_SMB)	8	R/W	00h	31.3.9/541
4006_6009	I2C Address Register 2 (I2C0_A2)	8	R/W	C2h	31.3.10/ 542
4006_600A	I2C SCL Low Timeout Register High (I2C0_SLTH)	8	R/W	00h	31.3.11/ 543
4006_600B	I2C SCL Low Timeout Register Low (I2C0_SLTL)	8	R/W	00h	31.3.12/ 543

31.3.1 I2C Address Register 1 (I2Cx_A1)

This register contains the slave address to be used by the I2C module.

Address: 4006_6000h base + 0h offset = 4006_6000h

Bit	7	6	5	4	3	2	1	0
Read	AD[7:1]							0
Write								▨
Reset	0	0	0	0	0	0	0	0

I2Cx_A1 field descriptions

Field	Description
7–1 AD[7:1]	Address Contains the primary slave address used by the I2C module when it is addressed as a slave. This field is used in the 7-bit address scheme and the lower seven bits in the 10-bit address scheme.

Table continues on the next page...

I2Cx_A1 field descriptions (continued)

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.2 I2C Frequency Divider register (I2Cx_F)

Address: 4006_6000h base + 1h offset = 4006_6001h

Bit	7	6	5	4	3	2	1	0
Read Write	MULT		ICR					
Reset	0	0	0	0	0	0	0	0

I2Cx_F field descriptions

Field	Description
7–6 MULT	Multiplier Factor Defines the multiplier factor (mul). This factor is used along with the SCL divider to generate the I2C baud rate. 00 mul = 1 01 mul = 2 10 mul = 4 11 Reserved

ICR	ClockRate
	
	Prescales the I2C module clock for bit rate selection. This field and the MULT field determine the I2C baud rate, the SDA hold time, the SCL start hold time, and the SCL stop hold time. For a list of values corresponding to each ICR setting, see I2C divider and hold values.
	
	The SCL divider multiplied by multiplier factor (mul) determines the I2C baud rate.
	
	`I2C baud rate = I2C module clock speed (Hz)/(mul × SCL divider)`
	
	The SDA hold time is the delay from the falling edge of SCL (I2C clock) to the changing of SDA (I2C data).
	
	`SDA hold time = I2C module clock period (s) × mul × SDA hold value`
	
	The SCL start hold time is the delay from the falling edge of SDA (I2C data) while SCL is high (start condition) to the falling edge of SCL (I2C clock).
	
	`SCL start hold time = I2C module clock period (s) × mul × SCL start hold value`
	
	The SCL stop hold time is the delay from the rising edge of SCL (I2C clock) to the rising edge of SDA (I2C data) while SCL is high (stop condition).
	
	`SCL stop hold time = I2C module clock period (s) × mul × SCL stop hold value`
	
	For example, if the I2C module clock speed is 8 MHz, the following table shows the possible hold time values with different ICR and MULT selections to achieve an I^2C baud rate of 100 kbit/s.

Table continues on the next page...

I2Cx_F field descriptions (continued)

Field	Description				
	MULT	**ICR**	**Hold times (μs)**		
			SDA	**SCL Start**	**SCL Stop**
	2h	00h	3.500	3.000	5.500
	1h	07h	2.500	4.000	5.250
	1h	0Bh	2.250	4.000	5.250
	0h	14h	2.125	4.250	5.125
	0h	18h	1.125	4.750	5.125

31.3.3 I2C Control Register 1 (I2Cx_C1)

Address: 4006_6000h base + 2h offset = 4006_6002h

Bit	7	6	5	4	3	2	1	0
Read	IICEN	IICIE	MST	TX	TXAK	0	WUEN	0
Write						RSTA		▨
Reset	0	0	0	0	0	0	0	0

I2Cx_C1 field descriptions

Field	Description
7 IICEN	I2C Enable Enables I2C module operation. 0 Disabled 1 Enabled
6 IICIE	I2C Interrupt Enable Enables I2C interrupt requests. 0 Disabled 1 Enabled
5 MST	Master Mode Select When MST is changed from 0 to 1, a START signal is generated on the bus and master mode is selected. When this bit changes from 1 to 0, a STOP signal is generated and the mode of operation changes from master to slave. 0 Slave mode 1 Master mode
4 TX	Transmit Mode Select

Table continues on the next page...

I2Cx_C1 field descriptions (continued)

Field	Description
	Selects the direction of master and slave transfers. In master mode this bit must be set according to the type of transfer required. Therefore, for address cycles, this bit is always set. When addressed as a slave this bit must be set by software according to the SRW bit in the status register. 0 Receive 1 Transmit
3 TXAK	Transmit Acknowledge Enable Specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. The value of SMB[FACK] affects NACK/ACK generation. **NOTE:** SCL is held low until TXAK is written. 0 An acknowledge signal is sent to the bus on the following receiving byte (if FACK is cleared) or the current receiving byte (if FACK is set). 1 No acknowledge signal is sent to the bus on the following receiving data byte (if FACK is cleared) or the current receiving data byte (if FACK is set).
2 RSTA	Repeat START Writing 1 to this bit generates a repeated START condition provided it is the current master. This bit will always be read as 0. Attempting a repeat at the wrong time results in loss of arbitration.
1 WUEN	Wakeup Enable The I2C module can wake the MCU from low power mode with no peripheral bus running when slave address matching occurs. 0 Normal operation. No interrupt generated when address matching in low power mode. 1 Enables the wakeup function in low power mode.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.4 I2C Status register (I2Cx_S)

Address: 4006_6000h base + 3h offset = 4006_6003h

Bit	7	6	5	4	3	2	1	0
Read	TCF	IAAS	BUSY	ARBL	RAM	SRW	IICIF	RXAK
Write	▨		▨	w1c		▨	w1c	▨
Reset	1	0	0	0	0	0	0	0

I2Cx_S field descriptions

Field	Description
7 TCF	Transfer Complete Flag Acknowledges a byte transfer; TCF sets on the completion of a byte transfer. This bit is valid only during or immediately following a transfer to or from the I2C module. TCF is cleared by reading the I2C data register in receive mode or by writing to the I2C data register in transmit mode.

Table continues on the next page...

I2Cx_S field descriptions (continued)

Field	Description
	0 Transfer in progress 1 Transfer complete
6 IAAS	Addressed As A Slave This bit is set by one of the following conditions: • The calling address matches the programmed primary slave address in the A1 register, or matches the range address in the RA register (which must be set to a nonzero value and under the condition I2C_C2[RMEN] = 1). • C2[GCAEN] is set and a general call is received. • SMB[SIICAEN] is set and the calling address matches the second programmed slave address. • ALERTEN is set and an SMBus alert response address is received • RMEN is set and an address is received that is within the range between the values of the A1 and RA registers. IAAS sets before the ACK bit. The CPU must check the SRW bit and set TX/RX accordingly. Writing the C1 register with any value clears this bit. 0 Not addressed 1 Addressed as a slave
5 BUSY	Bus Busy Indicates the status of the bus regardless of slave or master mode. This bit is set when a START signal is detected and cleared when a STOP signal is detected. 0 Bus is idle 1 Bus is busy
4 ARBL	Arbitration Lost This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software, by writing 1 to it. 0 Standard bus operation. 1 Loss of arbitration.
3 RAM	Range Address Match This bit is set to 1 by any of the following conditions, if I2C_C2[RMEN] = 1: • Any nonzero calling address is received that matches the address in the RA register. • The calling address is within the range of values of the A1 and RA registers. Writing the C1 register with any value clears this bit to 0. 0 Not addressed 1 Addressed as a slave
2 SRW	Slave Read/Write When addressed as a slave, SRW indicates the value of the R/$\overline{W}$ command bit of the calling address sent to the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IICIF	Interrupt Flag

Table continues on the next page...

I2Cx_S field descriptions (continued)

Field	Description
	This bit sets when an interrupt is pending. This bit must be cleared by software by writing 1 to it, such as in the interrupt routine. One of the following events can set this bit: • One byte transfer, including ACK/NACK bit, completes if FACK is 0. An ACK or NACK is sent on the bus by writing 0 or 1 to TXAK after this bit is set in receive mode. • One byte transfer, excluding ACK/NACK bit, completes if FACK is 1. • Match of slave address to calling address including primary slave address, range slave address, alert response address, second slave address, or general call address. • Arbitration lost • In SMBus mode, any timeouts except SCL and SDA high timeouts • I2C bus stop or start detection if the SSIE bit in the Input Glitch Filter register is 1 **NOTE:** To clear the I2C bus stop or start detection interrupt: In the interrupt service routine, first clear the STOPF or STARTF bit in the Input Glitch Filter register by writing 1 to it, and then clear the IICIF bit. If this sequence is reversed, the IICIF bit is asserted again. 0 No interrupt pending 1 Interrupt pending
0 RXAK	Receive Acknowledge 0 Acknowledge signal was received after the completion of one byte of data transmission on the bus 1 No acknowledge signal detected

31.3.5 I2C Data I/O register (I2Cx_D)

Address: 4006_6000h base + 4h offset = 4006_6004h

Bit	7	6	5	4	3	2	1	0
Read Write	DATA							
Reset	0	0	0	0	0	0	0	0

I2Cx_D field descriptions

Field	Description
DATA	Data In master transmit mode, when data is written to this register, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data. **NOTE:** When making the transition out of master receive mode, switch the I2C mode before reading the Data register to prevent an inadvertent initiation of a master receive data transfer. In slave mode, the same functions are available after an address match occurs. The C1[TX] bit must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For example, if the I2C module is configured for master transmit but a master receive is desired, reading the Data register does not initiate the receive. Reading the Data register returns the last byte received while the I2C module is configured in master receive or slave receive mode. The Data register does not reflect every byte that is transmitted on the I2C bus, and neither can software verify that a byte has been written to the Data register correctly by reading it back.

I2Cx_D field descriptions (continued)

Field	Description
	In master transmit mode, the first byte of data written to the Data register following assertion of MST (start bit) or assertion of RSTA (repeated start bit) is used for the address transfer and must consist of the calling address (in bits 7-1) concatenated with the required R/$\overline{W}$ bit (in position bit 0).

31.3.6 I2C Control Register 2 (I2Cx_C2)

Address: 4006_6000h base + 5h offset = 4006_6005h

Bit	7	6	5	4	3	2	1	0
Read	GCAEN	ADEXT	0	SBRC	RMEN	AD[10:8]		
Write			▨					
Reset	0	0	0	0	0	0	0	0

I2Cx_C2 field descriptions

Field	Description
7 GCAEN	General Call Address Enable Enables general call address. 0 Disabled 1 Enabled
6 ADEXT	Address Extension Controls the number of bits used for the slave address. 0 7-bit address scheme 1 10-bit address scheme
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 SBRC	Slave Baud Rate Control Enables independent slave mode baud rate at maximum frequency, which forces clock stretching on SCL in very fast I2C modes. To a slave, an example of a "very fast" mode is when the master transfers at 40 kbit/s but the slave can capture the master's data at only 10 kbit/s. 0 The slave baud rate follows the master baud rate and clock stretching may occur 1 Slave baud rate is independent of the master baud rate
3 RMEN	Range Address Matching Enable This bit controls the slave address matching for addresses between the values of the A1 and RA registers. When this bit is set, a slave address matching occurs for any address greater than the value of the A1 register and less than or equal to the value of the RA register. 0 Range mode disabled. No address matching occurs for an address within the range of values of the A1 and RA registers. 1 Range mode enabled. Address matching occurs when a slave receives an address within the range of values of the A1 and RA registers.
AD[10:8]	Slave Address

Table continues on the next page...

I2Cx_C2 field descriptions (continued)

Field	Description
	Contains the upper three bits of the slave address in the 10-bit address scheme. This field is valid only while the ADEXT bit is set.

31.3.7 I2C Programmable Input Glitch Filter register (I2Cx_FLT)

Address: 4006_6000h base + 6h offset = 4006_6006h

Bit	7	6	5	4	3	2	1	0
Read	SHEN	STOPF	SSIE	STARTF	FLT			
Write		w1c		w1c				
Reset	0	0	0	0	0	0	0	0

I2Cx_FLT field descriptions

Field	Description
7 SHEN	Stop Hold Enable Set this bit to hold off entry to stop mode when any data transmission or reception is occurring. The following scenario explains the holdoff functionality: 1. The I2C module is configured for a basic transfer, and the SHEN bit is set to 1. 2. A transfer begins. 3. The MCU signals the I2C module to enter stop mode. 4. The byte currently being transferred, including both address and data, completes its transfer. 5. The I2C slave or master acknowledges that the in-transfer byte completed its transfer and acknowledges the request to enter stop mode. 6. After receiving the I2C module's acknowledgment of the request to enter stop mode, the MCU determines whether to shut off the I2C module's clock. If the SHEN bit is set to 1 and the I2C module is in an idle or disabled state when the MCU signals to enter stop mode, the module immediately acknowledges the request to enter stop mode. If SHEN is cleared to 0 and the overall data transmission or reception that was suspended by stop mode entry was incomplete: To resume the overall transmission or reception after the MCU exits stop mode, software must reinitialize the transfer by resending the address of the slave. If the I2C Control Register 1's IICIE bit was set to 1 before the MCU entered stop mode, system software will receive the interrupt triggered by the I2C Status Register's TCF bit after the MCU wakes from the stop mode. 0 Stop holdoff is disabled. The MCU's entry to stop mode is not gated. 1 Stop holdoff is enabled.
6 STOPF	I2C Bus Stop Detect Flag Hardware sets this bit when the I2C bus's stop status is detected. The STOPF bit must be cleared by writing 1 to it. 0 No stop happens on I2C bus 1 Stop detected on I2C bus
5 SSIE	I2C Bus Stop or Start Interrupt Enable This bit enables the interrupt for I2C bus stop or start detection.

Table continues on the next page...

I2Cx_FLT field descriptions (continued)

Field	Description
	NOTE: To clear the I2C bus stop or start detection interrupt: In the interrupt service routine, first clear the STOPF or STARTF bit by writing 1 to it, and then clear the IICIF bit in the status register. If this sequence is reversed, the IICIF bit is asserted again. 0 Stop or start detection interrupt is disabled 1 Stop or start detection interrupt is enabled
4 STARTF	I2C Bus Start Detect Flag Hardware sets this bit when the I2C bus's start status is detected. The STARTF bit must be cleared by writing 1 to it. 0 No start happens on I2C bus 1 Start detected on I2C bus
FLT	I2C Programmable Filter Factor Controls the width of the glitch, in terms of I2C module clock cycles, that the filter must absorb. For any glitch whose size is less than or equal to this width setting, the filter does not allow the glitch to pass. 0h No filter/bypass 1-Fh Filter glitches up to width of *n* I2C module clock cycles, where *n*=1-15d

31.3.8 I2C Range Address register (I2Cx_RA)

Address: 4006_6000h base + 7h offset = 4006_6007h

Bit	7	6	5	4	3	2	1	0
Read	RAD							0
Write								▨
Reset	0	0	0	0	0	0	0	0

I2Cx_RA field descriptions

Field	Description
7–1 RAD	Range Slave Address This field contains the slave address to be used by the I2C module. The field is used in the 7-bit address scheme. If I2C_C2[RMEN] is set to 1, any nonzero value write enables this register. This register value can be considered as a maximum boundary in the range matching mode.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.9 I2C SMBus Control and Status register (I2Cx_SMB)

NOTE

When the SCL and SDA signals are held high for a length of time greater than the high timeout period, the SHTF1 flag sets. Before reaching this threshold, while the system is detecting how long these signals are being held high, a master assumes that the bus is free. However, the SHTF1 bit is set to 1 in the bus transmission process with the idle bus state.

NOTE

When the TCKSEL bit is set, there is no need to monitor the SHTF1 bit because the bus speed is too high to match the protocol of SMBus.

Address: 4006_6000h base + 8h offset = 4006_6008h

Bit	7	6	5	4	3	2	1	0
Read	FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF1	SHTF2	SHTF2IE
Write					w1c	▨	w1c	
Reset	0	0	0	0	0	0	0	0

I2Cx_SMB field descriptions

Field	Description
7 FACK	Fast NACK/ACK Enable For SMBus packet error checking, the CPU must be able to issue an ACK or NACK according to the result of receiving data byte. 0 An ACK or NACK is sent on the following receiving data byte 1 Writing 0 to TXAK after receiving a data byte generates an ACK. Writing 1 to TXAK after receiving a data byte generates a NACK.
6 ALERTEN	SMBus Alert Response Address Enable Enables or disables SMBus alert response address matching. **NOTE:** After the host responds to a device that used the alert response address, you must use software to put the device's address on the bus. The alert protocol is described in the SMBus specification. 0 SMBus alert response address matching is disabled 1 SMBus alert response address matching is enabled
5 SIICAEN	Second I2C Address Enable Enables or disables SMBus device default address. 0 I2C address register 2 matching is disabled 1 I2C address register 2 matching is enabled

Table continues on the next page...

I2Cx_SMB field descriptions (continued)

Field	Description
4 TCKSEL	Timeout Counter Clock Select Selects the clock source of the timeout counter. 0 Timeout counter counts at the frequency of the I2C module clock / 64 1 Timeout counter counts at the frequency of the I2C module clock
3 SLTF	SCL Low Timeout Flag This bit is set when the SLT register (consisting of the SLTH and SLTL registers) is loaded with a non-zero value (LoValue) and an SCL low timeout occurs. Software clears this bit by writing a logic 1 to it. **NOTE:** The low timeout function is disabled when the SLT register's value is 0. 0 No low timeout occurs 1 Low timeout occurs
2 SHTF1	SCL High Timeout Flag 1 This read-only bit sets when SCL and SDA are held high more than clock × LoValue / 512, which indicates the bus is free. This bit is cleared automatically. 0 No SCL high and SDA high timeout occurs 1 SCL high and SDA high timeout occurs
1 SHTF2	SCL High Timeout Flag 2 This bit sets when SCL is held high and SDA is held low more than clock × LoValue / 512. Software clears this bit by writing 1 to it. 0 No SCL high and SDA low timeout occurs 1 SCL high and SDA low timeout occurs
0 SHTF2IE	SHTF2 Interrupt Enable Enables SCL high and SDA low timeout interrupt. 0 SHTF2 interrupt is disabled 1 SHTF2 interrupt is enabled

31.3.10 I2C Address Register 2 (I2Cx_A2)

Address: 4006_6000h base + 9h offset = 4006_6009h

Bit	7	6	5	4	3	2	1	0
Read	SAD							0
Write								▨
Reset	1	1	0	0	0	0	1	0

I2Cx_A2 field descriptions

Field	Description
7–1 SAD	SMBus Address

Table continues on the next page...

I2Cx_A2 field descriptions (continued)

Field	Description
	Contains the slave address used by the SMBus. This field is used on the device default address or other related addresses.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.11 I2C SCL Low Timeout Register High (I2Cx_SLTH)

Address: 4006_6000h base + Ah offset = 4006_600Ah

Bit	7	6	5	4	3	2	1	0
Read Write	SSLT[15:8]							
Reset	0	0	0	0	0	0	0	0

I2Cx_SLTH field descriptions

Field	Description
SSLT[15:8]	Most significant byte of SCL low timeout value that determines the timeout period of SCL low.

31.3.12 I2C SCL Low Timeout Register Low (I2Cx_SLTL)

Address: 4006_6000h base + Bh offset = 4006_600Bh

Bit	7	6	5	4	3	2	1	0
Read Write	SSLT[7:0]							
Reset	0	0	0	0	0	0	0	0

I2Cx_SLTL field descriptions

Field	Description
SSLT[7:0]	Least significant byte of SCL low timeout value that determines the timeout period of SCL low.

31.4 Functional description

This section provides a comprehensive functional description of the I2C module.

31.4.1 I2C protocol

The I2C bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfers.

All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors depends on the system.

Normally, a standard instance of communication is composed of four parts:

1. START signal
2. Slave address transmission
3. Data transfer
4. STOP signal

The STOP signal should not be confused with the CPU STOP instruction. The following figure illustrates I2C bus system communication.

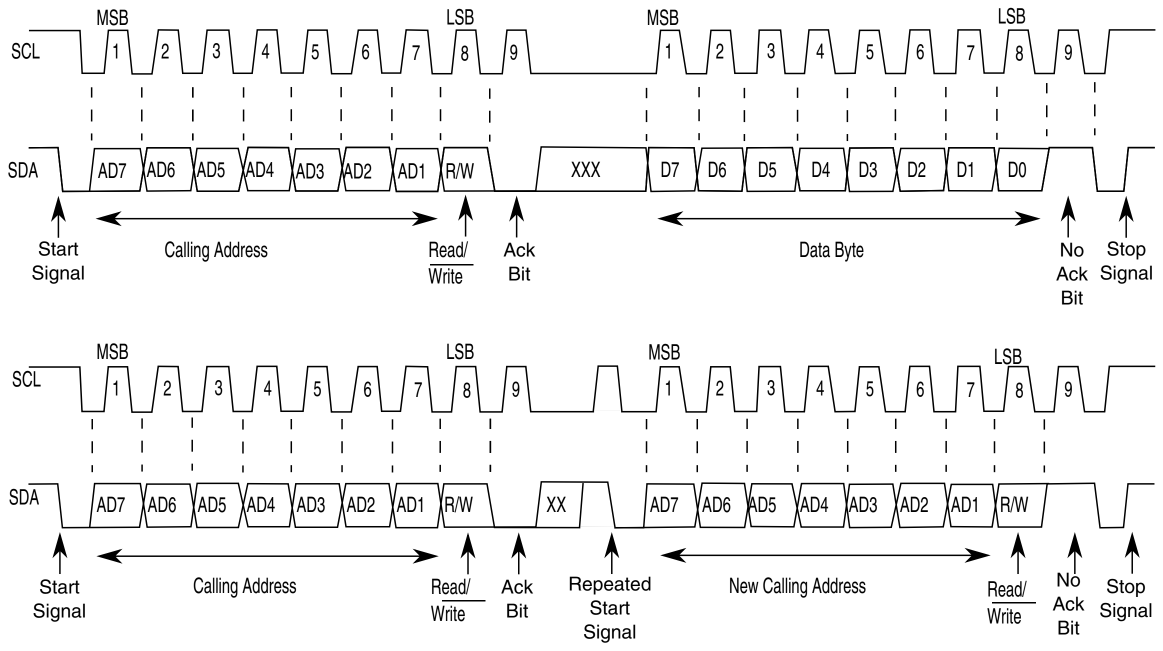


Figure 31-26. I2C bus transmission signals

31.4.1.1 START signal

The bus is free when no master device is engaging the bus (both SCL and SDA are high). When the bus is free, a master may initiate communication by sending a START signal. A START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer—each data transfer might contain several bytes of data—and brings all slaves out of their idle states.

31.4.1.2 Slave address transmission

Immediately after the START signal, the first byte of a data transfer is the slave address transmitted by the master. This address is a 7-bit calling address followed by an R/$\overline{\text{W}}$ bit. The R/$\overline{\text{W}}$ bit tells the slave the desired direction of data transfer.

- 1 = Read transfer: The slave transmits data to the master
- 0 = Write transfer: The master transmits data to the slave

Only the slave with a calling address that matches the one transmitted by the master responds by sending an acknowledge bit. The slave sends the acknowledge bit by pulling SDA low at the ninth clock.

No two slaves in the system can have the same address. If the I2C module is the master, it must not transmit an address that is equal to its own slave address. The I2C module cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the I2C module reverts to slave mode and operates correctly even if it is being addressed by another master.

31.4.1.3 Data transfers

When successful slave addressing is achieved, data transfer can proceed on a byte-by-byte basis in the direction specified by the R/$\overline{\text{W}}$ bit sent by the calling master.

All transfers that follow an address cycle are referred to as data transfers, even if they carry subaddress information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low. Data must be held stable while SCL is high. There is one clock pulse on SCL for each data bit, and the MSB is transferred first. Each data byte is followed by a ninth (acknowledge) bit, which is signaled from the receiving device by pulling SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the ninth bit, the slave must leave SDA high. The master interprets the failed acknowledgement as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets it as an end to data transfer and releases the SDA line.

In the case of a failed acknowledgement by either the slave or master, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a STOP signal.
- Commences a new call by generating a repeated START signal.

31.4.1.4 STOP signal

The master can terminate the communication by generating a STOP signal to free the bus. A STOP signal is defined as a low-to-high transition of SDA while SCL is asserted.

The master can generate a STOP signal even if the slave has generated an acknowledgement, at which point the slave must release the bus.

31.4.1.5 Repeated START signal

The master may generate a START signal followed by a calling command without generating a STOP signal first. This action is called a repeated START. The master uses a repeated START to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

31.4.1.6 Arbitration procedure

The I2C bus is a true multimaster bus that allows more than one master to be connected on it.

If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock. The bus clock's low period is equal to the longest clock low period, and the high period is equal to the shortest one among the masters.

The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic level 1 while another master transmits logic level 0. The losing masters immediately switch to slave receive mode and stop driving SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, hardware sets a status bit to indicate the loss of arbitration.

31.4.1.7 Clock synchronization

Because wire AND logic is performed on SCL, a high-to-low transition on SCL affects all devices connected on the bus. The devices start counting their low period and, after a device's clock has gone low, that device holds SCL low until the clock reaches its high state. However, the change of low to high in this device clock might not change the state of SCL if another device clock is still within its low period. Therefore, the synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time; see the following diagram. When all applicable devices have counted off their low period, the synchronized clock SCL is released and pulled high. Afterward there is no difference between the device clocks and the state of SCL, and all devices start counting their high periods. The first device to complete its high period pulls SCL low again.

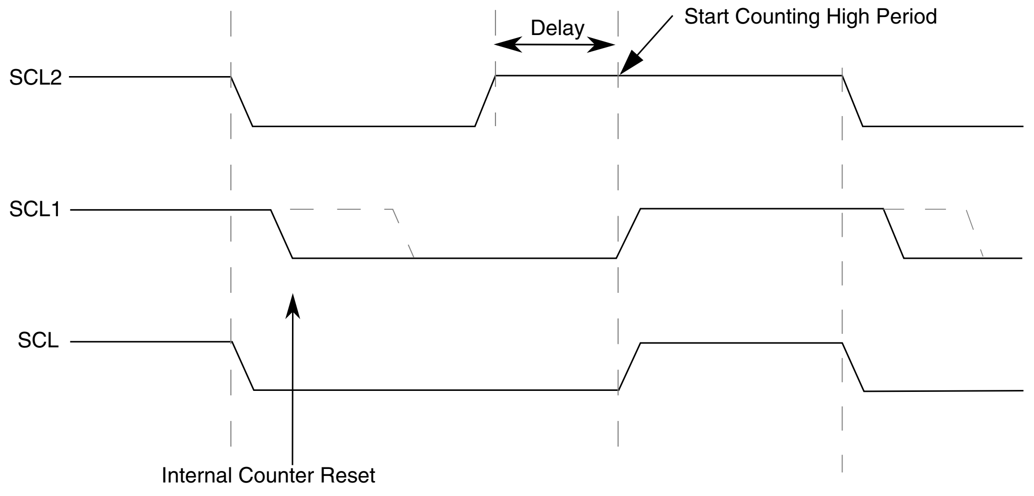


Figure 31-27. I2C clock synchronization

31.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfers. A slave device may hold SCL low after completing a single byte transfer (9 bits). In this case, it halts the bus clock and forces the master clock into wait states until the slave releases SCL.

31.4.1.9 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master drives SCL low, a slave can drive SCL low for the required period and then release it. If the slave's SCL low period is greater than the master's SCL low period, the resulting SCL bus signal's low period is stretched. In other words, the SCL bus signal's low period is increased to be the same length as the slave's SCL low period.

31.4.1.10 I2C divider and hold values

NOTE

For some cases on some devices, the SCL divider value may vary by ±2 or ±4 when ICR's value ranges from 00h to 0Fh. These potentially varying SCL divider values are highlighted in the following table. For the actual SCL divider values for your device, see the chip-specific details about the I2C module.

Table 31-28. I2C divider and hold values

ICR (hex)	SCL divider	SDA hold value	SCL hold (start) value	SCL hold (stop) value	ICR (hex)	SCL divider (clocks)	SDA hold (clocks)	SCL hold (start) value	SCL hold (stop) value
00	20	7	6	11	20	160	17	78	81
01	22	7	7	12	21	192	17	94	97
02	24	8	8	13	22	224	33	110	113
03	26	8	9	14	23	256	33	126	129
04	28	9	10	15	24	288	49	142	145
05	30	9	11	16	25	320	49	158	161
06	34	10	13	18	26	384	65	190	193
07	40	10	16	21	27	480	65	238	241
08	28	7	10	15	28	320	33	158	161
09	32	7	12	17	29	384	33	190	193
0A	36	9	14	19	2A	448	65	222	225
0B	40	9	16	21	2B	512	65	254	257
0C	44	11	18	23	2C	576	97	286	289
0D	48	11	20	25	2D	640	97	318	321
0E	56	13	24	29	2E	768	129	382	385
0F	68	13	30	35	2F	960	129	478	481
10	48	9	18	25	30	640	65	318	321
11	56	9	22	29	31	768	65	382	385
12	64	13	26	33	32	896	129	446	449
13	72	13	30	37	33	1024	129	510	513
14	80	17	34	41	34	1152	193	574	577
15	88	17	38	45	35	1280	193	638	641
16	104	21	46	53	36	1536	257	766	769
17	128	21	58	65	37	1920	257	958	961

Table continues on the next page...

Table 31-28. I2C divider and hold values (continued)

ICR (hex)	SCL divider	SDA hold value	SCL hold (start) value	SCL hold (stop) value		ICR (hex)	SCL divider (clocks)	SDA hold (clocks)	SCL hold (start) value	SCL hold (stop) value
18	80	9	38	41		38	1280	129	638	641
19	96	9	46	49		39	1536	129	766	769
1A	112	17	54	57		3A	1792	257	894	897
1B	128	17	62	65		3B	2048	257	1022	1025
1C	144	25	70	73		3C	2304	385	1150	1153
1D	160	25	78	81		3D	2560	385	1278	1281
1E	192	33	94	97		3E	3072	513	1534	1537
1F	240	33	118	121		3F	3840	513	1918	1921

31.4.2 10-bit address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

31.4.2.1 Master-transmitter addresses a slave-receiver

The transfer direction is not changed. When a 10-bit address follows a START condition, each slave compares the first 7 bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit ($R/\overline{W}$ direction bit) is 0. It is possible that more than one device finds a match and generates an acknowledge (A1). Each slave that finds a match compares the 8 bits of the second byte of the slave address with its own address, but only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

Table 31-29. Master-transmitter addresses slave-receiver with a 10-bit address

S	Slave address first 7 bits 11110 + AD10 + AD9	$R/\overline{W}$ 0	A1	Slave address second byte AD[8:1]	A2	Data	A	...	Data	A/A	P

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver sees an I2C interrupt. User software must ensure that for this interrupt, the contents of the Data register are ignored and not treated as valid data.

31.4.2.2 Master-receiver addresses a slave-transmitter

The transfer direction is changed after the second R/$\overline{\text{W}}$ bit. Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the START condition (S), and it tests whether the eighth (R/$\overline{\text{W}}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.

After a repeated START condition (Sr), all other slave devices also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{\text{W}}$) bit. However, none of them are addressed because R/$\overline{\text{W}}$ = 1 (for 10-bit devices), or the 11110XX slave address (for 7-bit devices) does not match.

Table 31-30. Master-receiver addresses a slave-transmitter with a 10-bit address

S	Slave address first 7 bits 11110 + AD10 + AD9	R/$\overline{\text{W}}$ 0	A1	Slave address second byte AD[8:1]	A2	Sr	Slave address first 7 bits 11110 + AD10 + AD9	R/$\overline{\text{W}}$ 1	A3	Data	A	...	Data	A	P

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter sees an I2C interrupt. User software must ensure that for this interrupt, the contents of the Data register are ignored and not treated as valid data.

31.4.3 Address matching

All received addresses can be requested in 7-bit or 10-bit address format.

- AD[7:1] in Address Register 1, which contains the I2C primary slave address, always participates in the address matching process. It provides a 7-bit address.
- If the ADEXT bit is set, AD[10:8] in Control Register 2 participates in the address matching process. It extends the I2C primary slave address to a 10-bit address.

Additional conditions that affect address matching include:

- If the GCAEN bit is set, general call participates the address matching process.
- If the ALERTEN bit is set, alert response participates the address matching process.
- If the SIICAEN bit is set, Address Register 2 participates in the address matching process.
- If the RMEN bit is set, when the Range Address register is programmed to a nonzero value, any address within the range of values of Address Register 1 (excluded) and the Range Address register (included) participates in the address matching process. The Range Address register must be programmed to a value greater than the value of Address Register 1.

When the I2C module responds to one of these addresses, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software must read the Data register after the first byte transfer to determine that the address is matched.

31.4.4 System management bus specification

SMBus provides a control bus for system and power management related tasks. A system can use SMBus to pass messages to and from devices instead of tripping individual control lines.

Removing the individual control lines reduces pin count. Accepting messages ensures future expandability. With the system management bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status.

31.4.4.1 Timeouts

The $T_{TIMEOUT,MIN}$ parameter allows a master or slave to conclude that a defective device is holding the clock low indefinitely or a master is intentionally trying to drive devices off the bus. The slave device must release the bus (stop driving the bus and let SCL and SDA float high) when it detects any single clock held low longer than $T_{TIMEOUT,MIN}$. Devices that have detected this condition must reset their communication and be able to receive a new START condition within the timeframe of $T_{TIMEOUT,MAX}$.

SMBus defines a clock low timeout, $T_{TIMEOUT}$, of 35 ms, specifies $T_{LOW:SEXT}$ as the cumulative clock low extend time for a slave device, and specifies $T_{LOW:MEXT}$ as the cumulative clock low extend time for a master device.

31.4.4.1.1 SCL low timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than a timeout value condition. Devices that have detected the timeout condition must reset the communication. When the I2C module is an active master, if it detects that SMBCLK low has exceeded the value of $T_{TIMEOUT,MIN}$, it must generate a stop condition within or after the current data byte in the transfer process. When the I2C module is a slave, if it detects the $T_{TIMEOUT,MIN}$ condition, it resets its communication and is then able to receive a new START condition.

31.4.4.1.2 SCL high timeout

When the I2C module has determined that the SMBCLK and SMBDAT signals have been high for at least $T_{HIGH:MAX}$, it assumes that the bus is idle.

A HIGH timeout occurs after a START condition appears on the bus but before a STOP condition appears on the bus. Any master detecting this scenario can assume the bus is free when either of the following occurs:

- SHTF1 rises.
- The BUSY bit is high and SHTF1 is high.

When the SMBDAT signal is low and the SMBCLK signal is high for a period of time, another kind of timeout occurs. The time period must be defined in software. SHTF2 is used as the flag when the time limit is reached. This flag is also an interrupt resource, so it triggers IICIF.

31.4.4.1.3 CSMBCLK TIMEOUT MEXT and CSMBCLK TIMEOUT SEXT

The following figure illustrates the definition of the timeout intervals $T_{LOW:SEXT}$ and $T_{LOW:MEXT}$. When in master mode, the I2C module must not cumulatively extend its clock cycles for a period greater than $T_{LOW:MEXT}$ within a byte, where each byte is defined as START-to-ACK, ACK-to-ACK, or ACK-to-STOP. When CSMBCLK TIMEOUT MEXT occurs, SMBus MEXT rises and also triggers the SLTF.

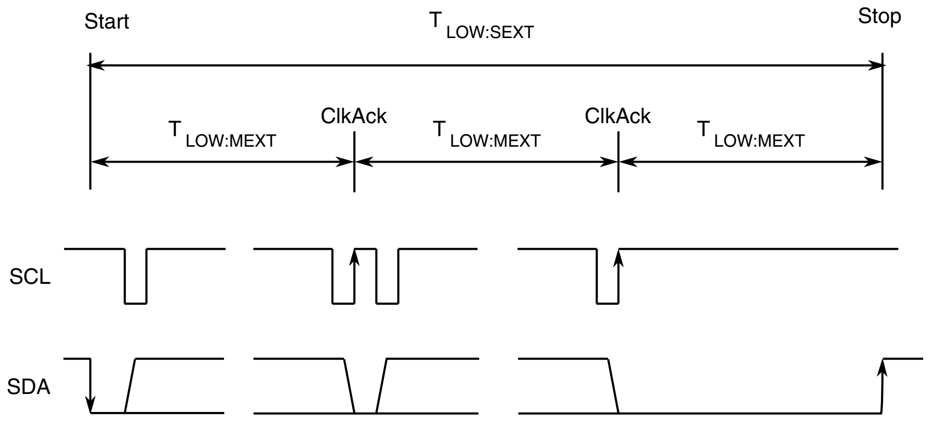


Figure 31-28. Timeout measurement intervals

A master is allowed to abort the transaction in progress to any slave that violates the $T_{LOW:SEXT}$ or $T_{TIMEOUT,MIN}$ specifications. To abort the transaction, the master issues a STOP condition at the conclusion of the byte transfer in progress. When a slave, the I2C module must not cumulatively extend its clock cycles for a period greater than $T_{LOW:SEXT}$ during any message from the initial START to the STOP. When CSMBCLK TIMEOUT SEXT occurs, SEXT rises and also triggers SLTF.

NOTE

CSMBCLK TIMEOUT SEXT and CSMBCLK TIMEOUT MEXT are optional functions that are implemented in the second step.

31.4.4.2 FAST ACK and NACK

To improve reliability and communication robustness, implementation of packet error checking (PEC) by SMBus devices is optional for SMBus devices but required for devices participating in and only during the address resolution protocol (ARP) process. The PEC is a CRC-8 error checking byte, calculated on all the message bytes. The PEC is appended to the message by the device that supplied the last data byte. If the PEC is present but not correct, a NACK is issued by the receiver. Otherwise an ACK is issued. To calculate the CRC-8 by software, this module can hold the SCL line low after receiving the eighth SCL (8th bit) if this byte is a data byte. So software can determine whether an ACK or NACK should be sent to the bus by setting or clearing the TXAK bit if the FACK (fast ACK/NACK enable) bit is enabled.

SMBus requires a device always to acknowledge its own address, as a mechanism to detect the presence of a removable device (such as a battery or docking station) on the bus. In addition to indicating a slave device busy condition, SMBus uses the NACK mechanism to indicate the reception of an invalid command or invalid data. Because such a condition may occur on the last byte of the transfer, SMBus devices are required to have the ability to generate the not acknowledge after the transfer of each byte and before the completion of the transaction. This requirement is important because SMBus does not provide any other resend signaling. This difference in the use of the NACK signaling has implications on the specific implementation of the SMBus port, especially in devices that handle critical system data such as the SMBus host and the SBS components.

NOTE

In the last byte of master receive slave transmit mode, the master must send a NACK to the bus, so FACK must be switched off before the last byte transmits.

31.4.5 Resets

The I2C module is disabled after a reset. The I2C module cannot cause a core reset.

31.4.6 Interrupts

The I2C module generates an interrupt when any of the events in the table found here occur, provided that the IICIE bit is set.

The interrupt is driven by the IICIF bit (of the I2C Status Register) and masked with the IICIE bit (of the I2C Control Register 1). The IICIF bit must be cleared (by software) by writing 1 to it in the interrupt routine. The SMBus timeouts interrupt is driven by SLTF and masked with the IICIE bit. The SLTF bit must be cleared by software by writing 1 to it in the interrupt routine. You can determine the interrupt type by reading the Status Register.

NOTE

In master receive mode, the FACK bit must be set to zero before the last byte transfer.

Table 31-31. Interrupt summary

Interrupt source	Status	Flag	Local enable
Complete 1-byte transfer	TCF	IICIF	IICIE
Match of received calling address	IAAS	IICIF	IICIE
Arbitration lost	ARBL	IICIF	IICIE
I^2C bus stop detection	STOPF	IICIF	IICIE & SSIE
I^2C bus start detection	STARTF	IICIF	IICIE & SSIE
SMBus SCL low timeout	SLTF	IICIF	IICIE
SMBus SCL high SDA low timeout	SHTF2	IICIF	IICIE & SHTF2IE
Wakeup from stop or wait mode	IAAS	IICIF	IICIE & WUEN

31.4.6.1 Byte transfer interrupt

The Transfer Complete Flag (TCF) bit is set at the falling edge of the ninth clock to indicate the completion of a byte and acknowledgement transfer. When FACK is enabled, TCF is then set at the falling edge of eighth clock to indicate the completion of byte.

31.4.6.2 Address detect interrupt

When the calling address matches the programmed slave address (I2C Address Register) or when the GCAEN bit is set and a general call is received, the IAAS bit in the Status Register is set. The CPU is interrupted, provided the IICIE bit is set. The CPU must check the SRW bit and set its Tx mode accordingly.

31.4.6.3 Stop Detect Interrupt

When the stop status is detected on the I^2C bus, the STOPF bit is set to 1. The CPU is interrupted, provided the IICIE and STOPIE bits are both set to 1.

31.4.6.4 Exit from low-power/stop modes

The slave receive input detect circuit and address matching feature are still active on low power modes (wait and stop). An asynchronous input matching slave address or general call address brings the CPU out of low power/stop mode if the interrupt is not masked. Therefore, TCF and IAAS both can trigger this interrupt.

31.4.6.5 Arbitration lost interrupt

The I2C is a true multimaster bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, the relative priority of the contending masters is determined by a data arbitration procedure. The I2C module asserts the arbitration-lost interrupt when it loses the data arbitration process and the ARBL bit in the Status Register is set.

Arbitration is lost in the following circumstances:

1. SDA is sampled as low when the master drives high during an address or data transmit cycle.

2. SDA is sampled as low when the master drives high during the acknowledge bit of a data receive cycle.
3. A START cycle is attempted when the bus is busy.
4. A repeated START cycle is requested in slave mode.
5. A STOP condition is detected when the master did not request it.

The ARBL bit must be cleared (by software) by writing 1 to it.

31.4.6.6 Timeout interrupt in SMBus

When the IICIE bit is set, the I2C module asserts a timeout interrupt (outputs SLTF and SHTF2) upon detection of any of the mentioned timeout conditions, with one exception. The SCL high and SDA high TIMEOUT mechanism must not be used to influence the timeout interrupt output, because this timeout indicates an idle condition on the bus. SHTF1 rises when it matches the SCL high and SDA high TIMEOUT and falls automatically just to indicate the bus status. The SHTF2's timeout period is the same as that of SHTF1, which is short compared to that of SLTF, so another control bit, SHTF2IE, is added to enable or disable it.

31.4.7 Programmable input glitch filter

An I2C glitch filter has been added outside legacy I2C modules but within the I2C package. This filter can absorb glitches on the I2C clock and data lines for the I2C module.

The width of the glitch to absorb can be specified in terms of the number of (half) I2C module clock cycles. A single Programmable Input Glitch Filter control register is provided. Effectively, any down-up-down or up-down-up transition on the data line that occurs within the number of clock cycles programmed in this register is ignored by the I2C module. The programmer must specify the size of the glitch (in terms of I2C module clock cycles) for the filter to absorb and not pass.

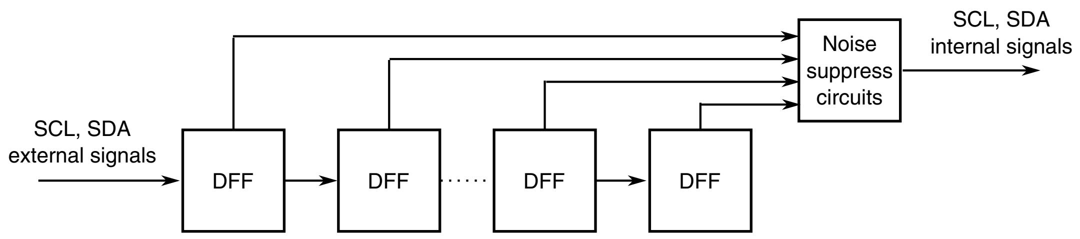


Figure 31-29. Programmable input glitch filter diagram

31.4.8 Address matching wake-up

When a primary, range, or general call address match occurs when the I2C module is in slave receive mode, the MCU wakes from a low power mode where no peripheral bus is running.

After the address matching IAAS bit is set, an interrupt is sent at the end of address matching to wake the core.

NOTE

During the wake-up process, if an external master continues to send data to the slave, the baud rate under Stop mode must be less than 50 kbit/s. To avoid the slower baud rate under Stop mode, the master can add a short delay in firmware to wait until the wake-up process is complete and then send data.

NOTE

Wake-up caused by an address match is not supported for SMBus mode.

31.5 Initialization/application information

Module Initialization (Slave)

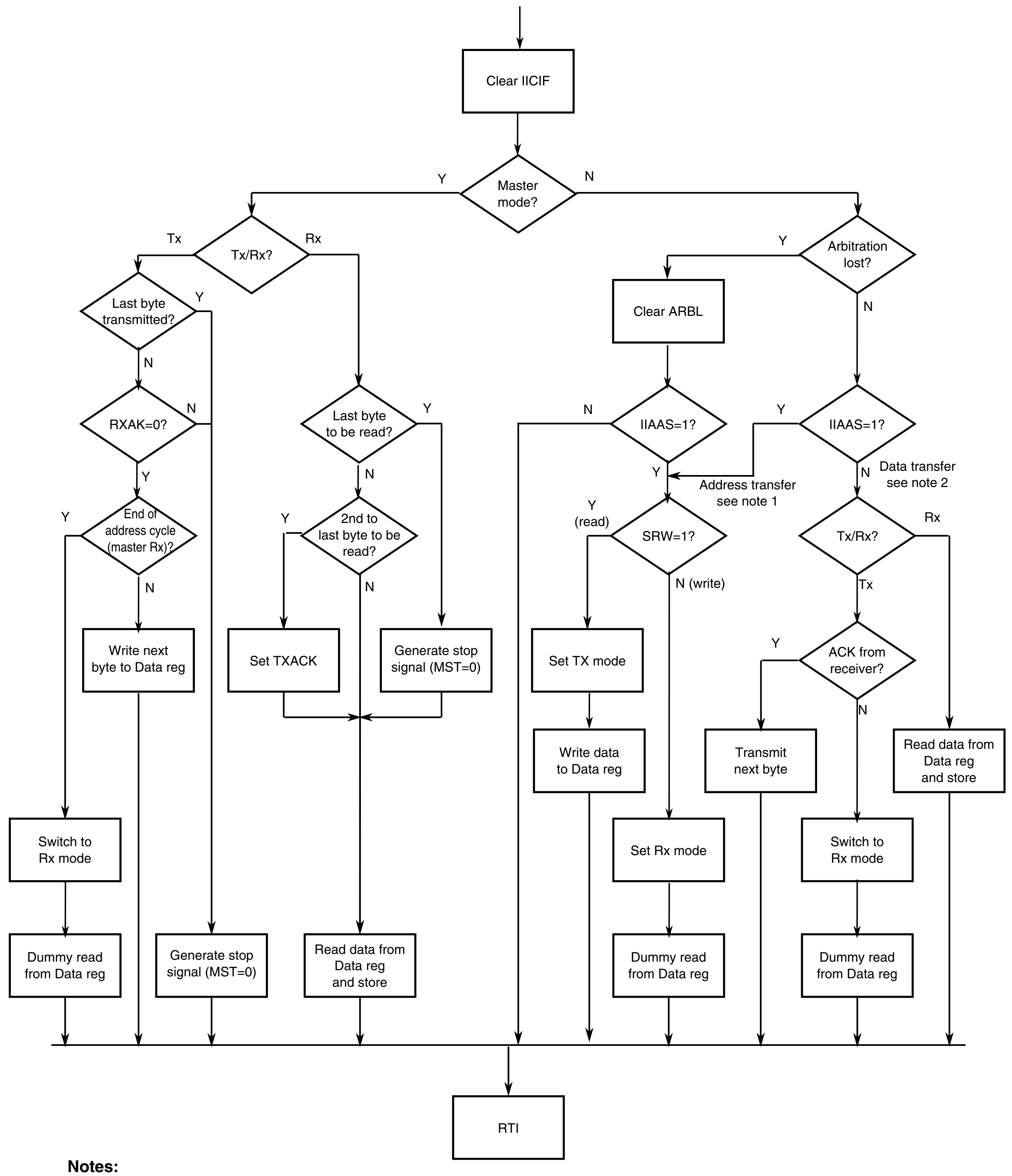
1. Write: Control Register 2
   - to enable or disable general call
   - to select 10-bit or 7-bit addressing mode
2. Write: Address Register 1 to set the slave address
3. Write: Control Register 1 to enable the I2C module and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in the following figure

Module Initialization (Master)

1. Write: Frequency Divider register to set the I2C baud rate (see example in description of ICR)
2. Write: Control Register 1 to enable the I2C module and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in the following figure
5. Write: Control Register 1 to enable TX

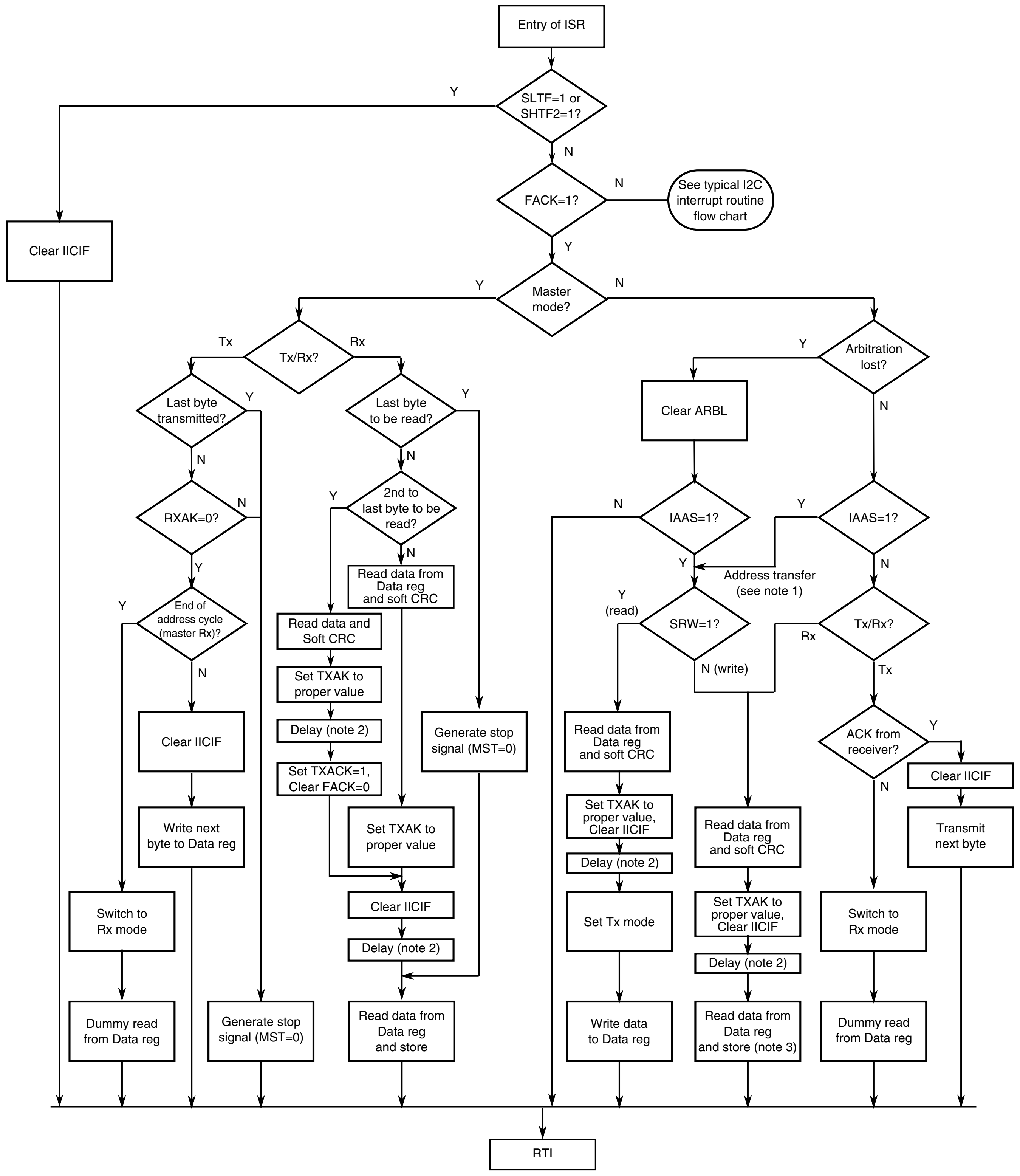
6. Write: Control Register 1 to enable MST (master mode)
7. Write: Data register with the address of the target slave (the LSB of this byte determines whether the communication is master receive or transmit)

The routine shown in the following figure encompasses both master and slave I2C operations. For slave operation, an incoming I2C message that contains the proper address begins I2C communication. For master operation, communication must be initiated by writing the Data register. An example of an I2C driver which implements many of the steps described here is available in AN4342: Using the Inter-Integrated Circuit on ColdFire+ and Kinetis .



Notes:
1. If general call is enabled, check to determine if the received address is a general call address (0x00). If the received address is a general call address, the general call must be handled by user software.
2. When 10-bit addressing addresses a slave, the slave sees an interrupt following the first byte of the extended address. Ensure that for this interrupt, the contents of the Data register are ignored and not treated as a valid data transfer.

Figure 31-30. Typical I2C interrupt routine



Notes:
1. If general call or SIICAEN is enabled, check to determine if the received address is a general call address (0x00) or an SMBus device default address. In either case, they must be handled by user software.
2. In receive mode, one bit time delay may be needed before the first and second data reading, to wait for the possible longest time period (in worst case) of the 9th SCL cycle.
3. This read is a dummy read in order to reset the SMBus receiver state machine.

Figure 31-31. Typical I2C SMBus interrupt routine

Chapter 32
Universal Asynchronous Receiver/Transmitter (UART)

32.1 Introduction

32.1.1 Features

Features of UART module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
    - Transmit data register empty and transmission complete
    - Receive data register full
    - Receive overrun, parity error, framing error, and noise error
    - Idle receiver detect
    - Active edge on receive pin
    - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Programmable 1-bit or 2-bit stop bits
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

32.1.2 Modes of operation

See Section Functional description for details concerning UART operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

32.1.3 Block diagram

The following figure shows the transmitter portion of the UART.

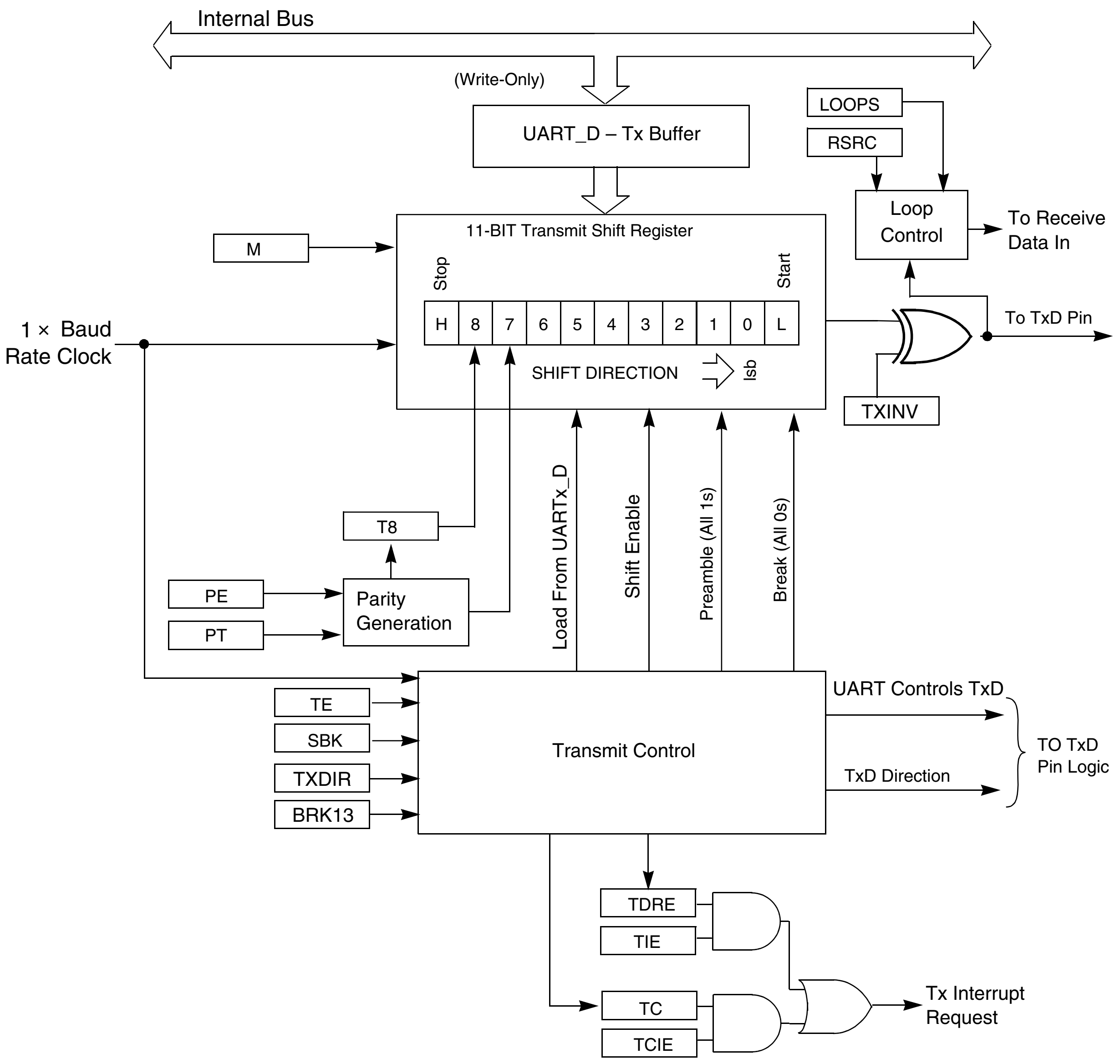


Figure 32-1. UART transmitter block diagram

The following figure shows the receiver portion of the UART.

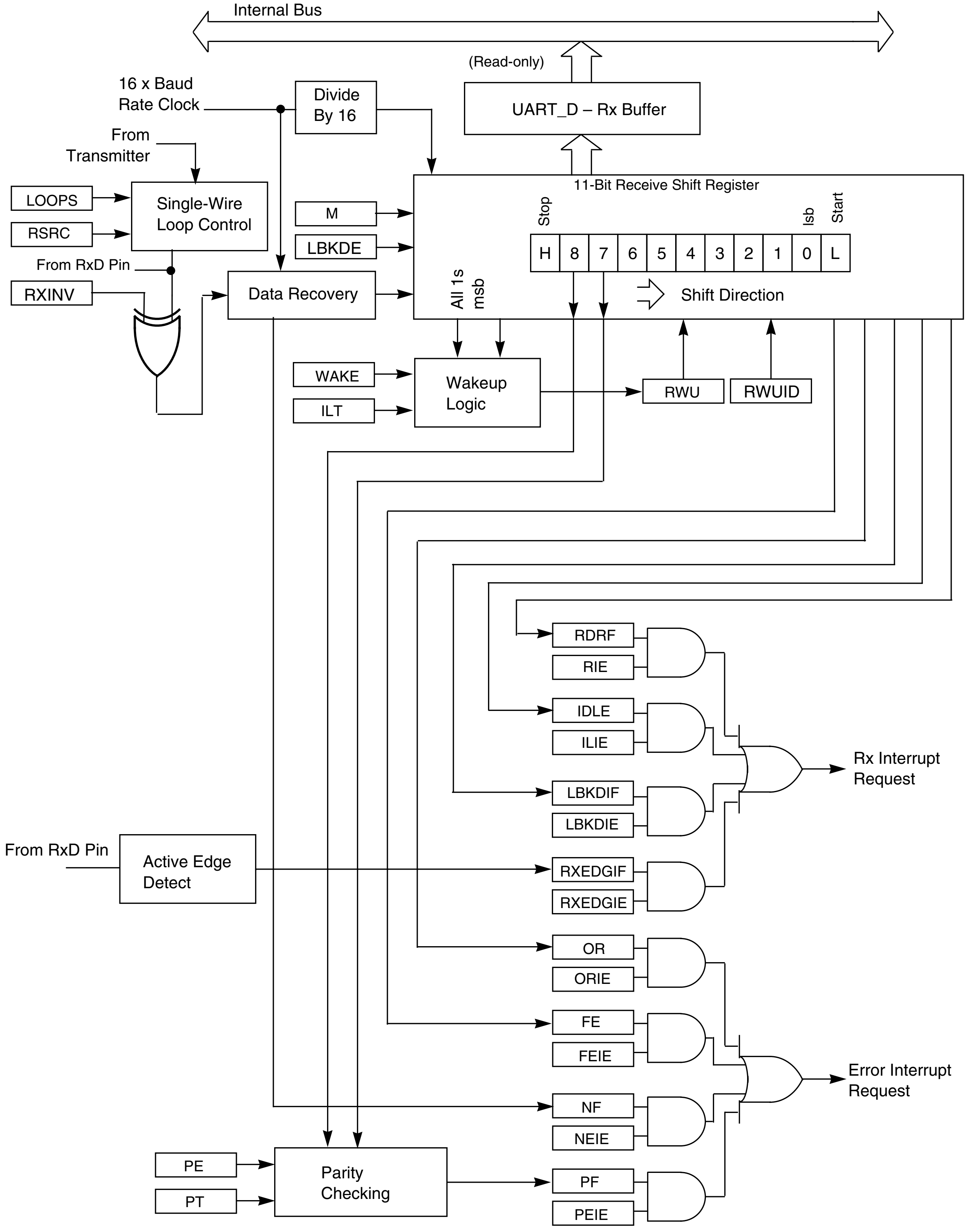


Figure 32-2. UART receiver block diagram

32.2 UART signal descriptions

The UART signals are shown in the table found here.

Table 32-1. UART signal descriptions

Signal	Description	I/O
RxD	Receive data	I
TxD	Transmit data	I/O

32.2.1 Detailed signal descriptions

The detailed signal descriptions of the UART are shown in the following table.

Table 32-2. UART—Detailed signal descriptions

Signal	I/O	Description	
RxD	I	Receive data. Serial data input to receiver.	
		State meaning	Whether RxD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.
		Timing	Sampled at a frequency determined by the module clock divided by the baud rate.
TxD	I/O	Transmit data. Serial data output from transmitter.	
		State meaning	Whether TxD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.
		Timing	Driven at the beginning or within a bit time according to the bit encoding method along with other configuration settings. Otherwise, transmissions are independent of reception timing.

32.3 Register definition

The UART has 8-bit registers to control baud rate, select UART options, report UART status, and for transmit/receive data.

Refer to the direct-page register summary in the memory chapter of this document or the absolute address assignments for all UART registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

UART memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4006_A000	UART Baud Rate Register: High (UART0_BDH)	8	R/W	00h	32.3.1/565
4006_A001	UART Baud Rate Register: Low (UART0_BDL)	8	R/W	04h	32.3.2/566
4006_A002	UART Control Register 1 (UART0_C1)	8	R/W	00h	32.3.3/566
4006_A003	UART Control Register 2 (UART0_C2)	8	R/W	00h	32.3.4/568
4006_A004	UART Status Register 1 (UART0_S1)	8	R	C0h	32.3.5/569
4006_A005	UART Status Register 2 (UART0_S2)	8	R/W	00h	32.3.6/571
4006_A006	UART Control Register 3 (UART0_C3)	8	R/W	00h	32.3.7/573
4006_A007	UART Data Register (UART0_D)	8	R/W	00h	32.3.8/574

32.3.1 UART Baud Rate Register: High (UARTx_BDH)

This register, along with UART_BDL, controls the prescale divisor for UART baud rate generation. To update the 13-bit baud rate setting SBR[12:0], first write to UART_BDH to buffer the high half of the new value and then write to UART_BDL. The working value in UART_BDH does not change until UART_BDL is written.

Address: 4006_A000h base + 0h offset = 4006_A000h

Bit	7	6	5	4	3	2	1	0
Read Write	LBKDIE	RXEDGIE	SBNS	SBR				
Reset	0	0	0	0	0	0	0	0

UARTx_BDH field descriptions

Field	Description
7 LBKDIE	LIN Break Detect Interrupt Enable (for LBKDIF) 0 Hardware interrupts from UART_S2[LBKDIF] disabled (use polling). 1 Hardware interrupt requested when UART_S2[LBKDIF] flag is 1.
6 RXEDGIE	RxD Input Active Edge Interrupt Enable (for RXEDGIF) 0 Hardware interrupts from UART_S2[RXEDGIF] disabled (use polling). 1 Hardware interrupt requested when UART_S2[RXEDGIF] flag is 1.
5 SBNS	Stop Bit Number Select SBNS determines whether data characters are one or two stop bits. 0 One stop bit. 1 Two stop bit.
SBR	Baud Rate Modulo Divisor.

Table continues on the next page...

UARTx_BDH field descriptions (continued)

Field	Description
	The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the UART baud rate generator. When BR is cleared, the UART baud rate generator is disabled to reduce supply current. When BR is 1 - 8191, the UART baud rate equals BUSCLK/(16×BR).

32.3.2 UART Baud Rate Register: Low (UARTx_BDL)

This register, along with UART_BDH, control the prescale divisor for UART baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to UART_BDH to buffer the high half of the new value and then write to UART_BDL. The working value in UART_BDH does not change until UART_BDL is written.

UART_BDL is reset to a non-zero value, so after reset, the baud rate generator remains disabled until the first time the receiver or transmitter is enabled; that is, 1 is written to UART_C2[RE] or UART_C2[TE].

Address: 4006_A000h base + 1h offset = 4006_A001h

Bit	7	6	5	4	3	2	1	0
Read / Write	SBR							
Reset	0	0	0	0	0	1	0	0

UARTx_BDL field descriptions

Field	Description
SBR	Baud Rate Modulo Divisor These 13 bits in SBR[12:0] are referred to collectively as BR, which set the modulo divide rate for the UART baud rate generator. When BR is cleared, the UART baud rate generator is disabled to reduce supply current. When BR is 1 - 8191, the UART baud rate equals BUSCLK/(16×BR).

32.3.3 UART Control Register 1 (UARTx_C1)

This read/write register controls various optional features of the UART system.

Address: 4006_A000h base + 2h offset = 4006_A002h

Bit	7	6	5	4	3	2	1	0
Read / Write	LOOPS	UARTSWAI	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

UARTx_C1 field descriptions

Field	Description
7 LOOPS	Loop Mode Select Selects between loop mode and normal 2-pin full-duplex modes. When LOOPS is set, the transmitter output is internally connected to the receiver input. 0 Normal operation - RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See RSRC bit.) RxD pin is not used by UART.
6 UARTSWAI	UART Stops in Wait Mode 0 UART clocks continue to run in Wait mode so the UART can be the source of an interrupt that wakes up the CPU. 1 UART clocks freeze while CPU is in Wait mode.
5 RSRC	Receiver Source Select This field has no meaning or effect unless LOOPS is set to 1. When LOOPS is set, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the UART does not use the RxD pins. 1 Single-wire UART mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select This field configures the UART to be operated in 9-bit or 8-bit data mode. 0 Normal - start + 8 data bits (lsb first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (lsb first) + 9th data bit + stop.
3 WAKE	Receiver Wakeup Method Select This field selects the receiver wakeup method. 0 Idle-line wake-up. 1 Address-mark wake-up.
2 ILT	Idle Line Type Select Setting this field to 1 ensures that the stop bits and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.
1 PE	Parity Enable Enables hardware parity generation and checking. When parity is enabled, the most significant bit (msb) of the data character, eighth or ninth data bit, is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	Parity Type

Table continues on the next page...

UARTx_C1 field descriptions (continued)

Field	Description
	Provided parity is enabled (PE = 1), this field selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

32.3.4 UART Control Register 2 (UARTx_C2)

This register can be read or written at any time.

Address: 4006_A000h base + 3h offset = 4006_A003h

Bit	7	6	5	4	3	2	1	0
Read Write	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

UARTx_C2 field descriptions

Field	Description
7 TIE	Transmit Interrupt Enable for TDRE 0 Hardware interrupts from TDRE disabled; use polling. 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	Transmission Complete Interrupt Enable for TC 0 Hardware interrupts from TC disabled; use polling. 1 Hardware interrupt requested when TC flag is 1.
5 RIE	Receiver Interrupt Enable for RDRF 0 Hardware interrupts from S1[RDRF] disabled; use polling. 1 Hardware interrupt requested when S1[RDRF] flag is 1.
4 ILIE	Idle Line Interrupt Enable for IDLE 0 Hardware interrupts from S1[IDLE] disabled; use polling. 1 Hardware interrupt requested when S1[IDLE] flag is 1.
3 TE	Transmitter Enable TE must be 1 to use the UART transmitter. When TE is set, the UART forces the TxD pin to act as an output for the UART system. When the UART is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single UART communication line (TxD pin). TE can also queue an idle character by clearing TE and then setting TE while a transmission is in progress. When 0 is written to TE, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin.

Table continues on the next page...

UARTx_C2 field descriptions (continued)

Field	Description
	0 Transmitter off. 1 Transmitter on.
2 RE	Receiver Enable When the UART receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If C1[LOOPS] is set, the RxD pin reverts to being a general-purpose I/O pin even if RE is set. 0 Receiver off. 1 Receiver on.
1 RWU	Receiver Wakeup Control A 1 can be written to this field to place the UART receiver in a standby state where it waits for automatic hardware detection of a selected wake-up condition. The wake-up condition is an idle line between messages, WAKE = 0, idle-line wake-up, or a logic 1 in the most significant data bit in a character, WAKE = 1, address-mark wake-up. Application software sets RWU and, normally, a selected hardware condition automatically clears RWU. 0 Normal UART receiver operation. 1 UART receiver in standby waiting for wake-up condition.
0 SBK	Send Break Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 or 12, 13 or 14 or 15 if BRK13 = 1, bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

32.3.5 UART Status Register 1 (UARTx_S1)

This register has eight read-only status flags. Writes have no effect. Special software sequences, which do not involve writing to this register, clear these status flags.

Address: 4006_A000h base + 4h offset = 4006_A004h

Bit	7	6	5	4	3	2	1	0
Read	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write								
Reset	1	1	0	0	0	0	0	0

UARTx_S1 field descriptions

Field	Description
7 TDRE	Transmit Data Register Empty Flag TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read UART_S1 with TDRE set and then write to the UART data register (UART_D).

Table continues on the next page...

UARTx_S1 field descriptions (continued)

Field	Description
	0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	Transmission Complete Flag TC is set out of reset and when TDRE is set and no data, preamble, or break character is being transmitted. TC is cleared automatically by reading UART_S1 with TC set and then executing one of the following operations: • Write to the UART data register (UART_D) to transmit new data • Queue a preamble by changing TE from 0 to 1 • Queue a break character by writing 1 to UART_C2[SBK] 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete).
5 RDRF	Receive Data Register Full Flag RDRF becomes set when a character transfers from the receive shifter into the receive data register (UART_D). To clear RDRF, read UART_S1 with RDRF set and then read the UART data register (UART_D). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	Idle Line Flag IDLE is set when the UART receive line becomes idle for a full character time after a period of activity. When C1[ILT] is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bits time count toward the full character time of logic high, 10 or 11 bit times depending on the M control bit, needed for the receiver to detect an idle line. When ILT is set, the receiver doesn't start counting idle bit times until the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, read UART_S1 with IDLE set and then read the UART data register (UART_D). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE is set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.
3 OR	Receiver Overrun Flag OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from UART_D yet. In this case, the new character, and all associated error information, is lost because there is no room to move it into UART_D. To clear OR, read UART_S1 with OR set and then read the UART data register (UART_D). 0 No overrun. 1 Receive overrun (new UART data lost).
2 NF	Noise Flag The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bits. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF is set at the same time as RDRF is set for the character. To clear NF, read UART_S1 and then read the UART data register (UART_D).

Table continues on the next page...

UARTx_S1 field descriptions (continued)

Field	Description
	0 No noise detected. 1 Noise detected in the received character in UART_D.
1 FE	Framing Error Flag FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bits were expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read UART_S1 with FE set and then read the UART data register (UART_D). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	Parity Error Flag PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read UART_S1 and then read the UART data register (UART_D). 0 No parity error. 1 Parity error.

32.3.6 UART Status Register 2 (UARTx_S2)

This register contains one read-only status flag.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave running 14% faster than the master. This would trigger normal break detection circuitry designed to detect a 10-bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

Address: 4006_A000h base + 5h offset = 4006_A005h

Bit	7	6	5	4	3	2	1	0
Read	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
Write			▨					▨
Reset	0	0	0	0	0	0	0	0

UARTx_S2 field descriptions

Field	Description
7 LBKDIF	LIN Break Detect Interrupt Flag LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a 1 to it.

Table continues on the next page...

UARTx_S2 field descriptions (continued)

Field	Description
	0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	RxD Pin Active Edge Interrupt Flag RXEDGIF is set when an active edge, falling if RXINV = 0, rising if RXINV=1, on the RxD pin occurs. RXEDGIF is cleared by writing a 1 to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 RXINV	Receive Data Inversion Setting this field reverses the polarity of the received data input. **NOTE:** Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle. 0 Receive data not inverted. 1 Receive data inverted.
3 RWUID	Receive Wake Up Idle Detect RWUID controls whether the idle character that wakes up the receiver sets S1[IDLE]. 0 During receive standby state (RWU = 1), S1[IDLE] does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), S1[IDLE] gets set upon detection of an idle character.
2 BRK13	Break Character Generation Length BRK13 selects a longer transmitted break character length. Detection of a framing error is not affected by the state of this field. 0 Break character is transmitted with length of 10 bit times (if M = 0, SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1, SBNS = 1). 1 Break character is transmitted with length of 13 bit times (if M = 0, SBNS = 0) or 14 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 15 (if M = 1, SBNS = 1).
1 LBKDE	LIN Break Detection Enable LBKDE enables the break detection. While LBKDE is set, S1[FE] and S1[RDRF] flags are prevented from setting. 0 Break detection is disabled. 1 Break detection is enabled (Break character is detected at length 11 bit times (if C1[M] = 0, BDH[SBNS] = 0) or 12 (if C1[M] = 1, BDH[SBNS] = 0 or C1[M] = 0, BDH[SBNS] = 1) or 13 (if C1[M] = 1, BDH[SBNS] = 1)).
0 RAF	Receiver Active Flag RAF is set when the UART receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an UART character is being received before instructing the MCU to go to stop mode. 0 UART receiver idle waiting for a start bit. 1 UART receiver active (RxD input not idle).

32.3.7 UART Control Register 3 (UARTx_C3)

Address: 4006_A000h base + 6h offset = 4006_A006h

Bit	7	6	5	4	3	2	1	0
Read	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
Write								
Reset	0	0	0	0	0	0	0	0

UARTx_C3 field descriptions

Field	Description
7 R8	Ninth Data Bit for Receiver When the UART is configured for 9-bit data (C1[M] = 1), R8 can be thought of as a ninth receive data bit to the left of the msb of the buffered data in the UART_D register. When reading 9-bit data, read R8 before reading UART_D because reading UART_D completes automatic flag clearing sequences that could allow R8 and UART_D to be overwritten with new data.
6 T8	Ninth Data Bit for Transmitter When the UART is configured for 9-bit data (C1[M] = 1), T8 may be thought of as a ninth transmit data bit to the left of the msb of the data in the UART_D register. When writing 9-bit data, the entire 9-bit value is transferred to the UART shift register after UART_D is written so T8 should be written, if it needs to change from its previous value, before UART_D is written. If T8 does not need to change in the new value, such as when it is used to generate mark or space parity, it need not be written each time UART_D is written.
5 TXDIR	TxD Pin Direction in Single-Wire Mode When the UART is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this field determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.
4 TXINV	Transmit Data Inversion Setting this field reverses the polarity of the transmitted data output. **NOTE:** Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle. 0 Transmit data not inverted. 1 Transmit data inverted.
3 ORIE	Overrun Interrupt Enable Enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled; use polling. 1 Hardware interrupt requested when OR is set.
2 NEIE	Noise Error Interrupt Enable Enables the noise flag (NF) to generate hardware interrupt requests.

Table continues on the next page...

UARTx_C3 field descriptions (continued)

Field	Description
	0 NF interrupts disabled; use polling). 1 Hardware interrupt requested when NF is set.
1 FEIE	Framing Error Interrupt Enable Enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled; use polling). 1 Hardware interrupt requested when FE is set.
0 PEIE	Parity Error Interrupt Enable Enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled; use polling). 1 Hardware interrupt requested when PF is set.

32.3.8 UART Data Register (UARTx_D)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the UART status flags.

Address: 4006_A000h base + 7h offset = 4006_A007h

Bit	7	6	5	4	3	2	1	0
Read Write	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	R0T0
Reset	0	0	0	0	0	0	0	0

UARTx_D field descriptions

Field	Description
7 R7T7	Read receive data buffer 7 or write transmit data buffer 7.
6 R6T6	Read receive data buffer 6 or write transmit data buffer 6.
5 R5T5	Read receive data buffer 5 or write transmit data buffer 5.
4 R4T4	Read receive data buffer 4 or write transmit data buffer 4.
3 R3T3	Read receive data buffer 3 or write transmit data buffer 3.
2 R2T2	Read receive data buffer 2 or write transmit data buffer 2.

Table continues on the next page...

UARTx_D field descriptions (continued)

Field	Description
1 R1T1	Read receive data buffer 1 or write transmit data buffer 1.
0 R0T0	Read receive data buffer 0 or write transmit data buffer 0.

32.4 Functional description

The UART allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs.

The UART comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the UART, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the UART.

32.4.1 Baud rate generation

As shown in the figure found here, the clock source for the UART baud rate generator is the bus-rate clock.

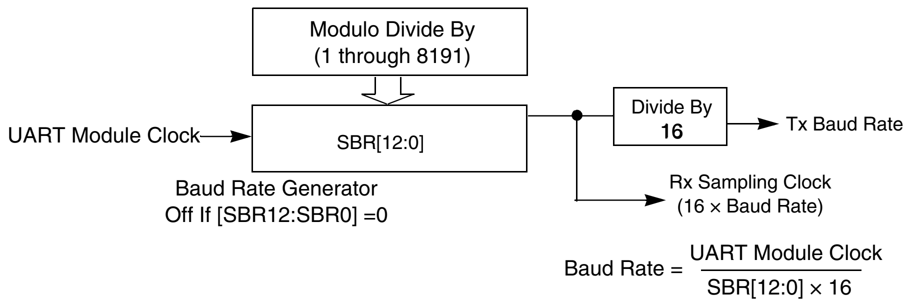


Figure 32-19. UART baud rate generation

UART communications require the transmitter and receiver, which typically derive baud rates from independent clock sources, to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition. In the worst case, there are no such transitions in the full 10- or 11-bit or 12-bittime character frame so any mismatch in baud rate is accumulated for the whole character time. For a

Freescale UART system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about ±4.5 percent for 8-bit data format and about ±4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

32.4.2 Transmitter functional description

This section describes the overall block diagram for the UART transmitter, as well as specialized functions for sending break and idle characters.

The transmitter output (TxD) idle state defaults to logic high, UART_C3[TXINV] is cleared following reset. The transmitter output is inverted by setting UART_C3[TXINV]. The transmitter is enabled by setting the TE bit in UART_C2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the UART data register (UART_D).

The central element of the UART transmitter is the transmit shift register that is 10 or 11 or 12 bits long depending on the setting in the UART_C1[M] control bit and UART_BDH[SBNS] bit. For the remainder of this section, assume UART_C1[M] is cleared, UART_BDH[SBNS] is also cleared, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new UART character, the value waiting in the transmit data register is transferred to the shift register, synchronized with the baud rate clock, and the transmit data register empty (UART_S1[TDRE]) status flag is set to indicate another character may be written to the transmit data buffer at UART_D.

NOTE

Always read UART_S1 before writing to UART_D to allow data to be transmitted.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to UART_C2[TE] does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

32.4.2.1 Send break and queued idle

UART_C2[SBK] sends break characters originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0, 10 bit times including the start and stop bits. A longer break of 13 bit times can be enabled by setting UART_S2[BRK13]. Normally, a program would wait for UART_S1[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, write 1, and then write 0 to UART_C2[SBK]. This action queues a break character to be sent as soon as the shifter is available. If UART_C2[SBK] remains 1 when the queued break moves into the shifter, synchronized to the baud rate clock, an additional break character is queued. If the receiving device is another Freescale Semiconductor UART, the break characters are received as 0s in all eight data bits and a framing error (UART_S1[FE] = 1) occurs.

When idle-line wake-up is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for UART_S1[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the UART_C2[TE] bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while UART_C2[TE] is cleared, the UART transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while UART_C2[TE] is cleared, set the general-purpose I/O controls so the pin shared with TxD is an output driving a logic 1. This ensures that the TxD line looks like a normal idle line even if the UART loses control of the port pin between writing 0 and then 1 to UART_C2[TE].

The length of the break character is affected by the UART_S2[BRK13] and UART_C1[M] as shown below.

Table 32-21. Break character length

BRK13	M	SBNS	Break character length
0	0	0	10 bit times
0	0	1	11 bit times
0	1	0	11 bit times
0	1	1	12 bit times
1	0	0	13 bit times
1	0	1	14 bit times
1	1	0	14 bit times
1	1	1	15 bit times

32.4.3 Receiver functional description

In this section, the receiver block diagram is a guide for the overall receiver functional description.

Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting UART_S2[RXINV]. The receiver is enabled by setting the UART_C2[RE] bit. Character frames consist of a start bit of logic 0, eight (or nine) data bits (lsb first), and one (or two) stop bits of logic 1. For information about 9-bit data mode, refer to 8- and 9-bit data modes. For the remainder of this discussion, assume the UART is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (UART_S1[RDRF]) status flag is set. If UART_S1[RDRF] was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the UART receiver is double-buffered, the program has one full character time after UART_S1[RDRF] is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (UART_S1[RDRF] = 1), it gets the data from the receive data register by reading UART_D. The UART_S1[RDRF] flag is cleared automatically by a two-step sequence normally satisfied in the course of the user's program that manages receive data. Refer to Interrupts and status flags for more details about flag clearing.

32.4.3.1 Data sampling technique

The UART receiver uses a 16× baud rate clock for sampling. The oversampling ratio is fixed at 16. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock divides the bit time into 16 segments labeled UART_D[RT1] through UART_D[RT16]. When a falling edge is located, three more samples are taken at UART_D[RT3], UART_D[RT5], and UART_D[RT7] to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at UART_D[RT8], UART_D[RT9], and UART_D[RT10] to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at UART_D[RT3], UART_D[RT5], and UART_D[RT7] are 0 even if one or all of the samples taken at UART_D[RT8], UART_D[RT9], and UART_D[RT10] are 1s. If any sample in any bit time, including the start and stop bits, in a character frame fails to agree with the logic level for that bit, the noise flag (UART_S1[NF]) is set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges. If an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if UART_S1[FE] remains set.

32.4.3.2 Receiver wake-up operation

Receiver wake-up is a hardware mechanism that allows an UART receiver to ignore the characters in a message intended for a different UART receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up control field (UART_C2[RWU]). When UART_C2[RWU] is set, the status flags associated with the receiver, (with the exception of the idle bit, IDLE, when UART_S2[RWUID] is set), are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force UART_C2[RWU] to 0, so all receivers wake up in time to look at the first character(s) of the next message.

32.4.3.2.1 Idle-line wakeup

When wake is cleared, the receiver is configured for idle-line wakeup. In this mode, UART_C2[RWU] is cleared automatically when the receiver detects a full character time of the idle-line level. The UART_C1[M] control field selects 8-bit or 9-bit data mode and

UART_BDH[SBNS] selects 1-bit or 2-bit stop bit number that determines how many bit times of idle are needed to constitute a full character time, 10 or 11 or 12 bit times because of the start and stop bits.

When UARTI_C2[RWU] is 1 and UART_S2[RWUID] is 0, the idle condition that wakes up the receiver does not set UART_S1[IDLE]. The receiver wakes up and waits for the first data character of the next message that sets UART_S1[RDRF] and generates an interrupt, if enabled. When UART_S2[RWUID] is 1, any idle condition sets UART_S1[IDLE] flag and generates an interrupt if enabled, regardless of whether UART_C2[RWU] is 0 or 1.

The idle-line type (UART_C1[ILT]) control bit selects one of two ways to detect an idle line. When UART_C1[ILT] is cleared, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When UART_C1[ILT] is set, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

32.4.3.2.2 Address-mark wakeup

When wake is set, the receiver is configured for address-mark wakeup. In this mode, UART_C2[RWU] is cleared automatically when the receiver detects a, or two, if UART_BDH[SBNS] = 1, logic 1 in the most significant bits of a received character, eighth bit when UART_C1[M] is cleared and ninth bit when UART_C1[M] is set.

Address-mark wakeup allows messages to contain idle characters, but requires the msb be reserved for use in address frames. The one, or two, if UART_BDH[SBNS] = 1, logic 1s msb of an address frame clears the UART_C2[RWU] bit before the stop bits are received and sets the UART_S1[RDRF] flag. In this case, the character with the msb set is received even though the receiver was sleeping during most of this character time.

32.4.4 Interrupts and status flags

The UART system has one interrupt vector which has three separate interrupt types. One interrupt type is associated with the transmitter for UART_S1[TDRE] and UART_S1[TC] events. Another interrupt type is associated with the receiver for RDRF, IDLE, RXEDGIF, and LBKDIF events. A third type is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The UART transmitter has two status flags that can optionally generate hardware interrupt requests. Transmit data register empty (UART_S1[TDRE]) indicates when there is room in the transmit data buffer to write another transmit character to UART_D. If the transmit interrupt enable (UART_C2[TIE]) bit is set, a hardware interrupt is requested when UART_S1[TDRE] is set. Transmit complete (UART_S1[TC]) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (UART_C2[TCIE]) bit is set, a hardware interrupt is requested when UART_S1[TC] is set. Instead of hardware interrupts, software polling may be used to monitor the UART_S1[TDRE] and UART_S1[TC] status flags if the corresponding UART_C2[TIE] or UART_C2[TCIE] local interrupt masks are cleared.

When a program detects that the receive data register is full (UART_S1[RDRF] = 1), it gets the data from the receive data register by reading UART_D. The UART_S1[RDRF] flag is cleared by reading UART_S1 while UART_S1[RDRF] is set and then reading UART_D.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, UART_S1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading UART_S1 while UART_S1[IDLE] is set and then reading UART_D. After UART_S1[IDLE] has been cleared, it cannot become set again until the receiver has received at least one new character and has set UART_S1[RDRF].

If the associated error was detected in the received character that caused UART_S1[RDRF] to be set, the error flags - noise flag (UART_S1[NF]), framing error (UART_S1[FE]), and parity error flag (UART_S1[PF]) - are set at the same time as UART_S1[RDRF]. These flags are not set in overrun cases.

If UART_S1[RDRF] was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (UART_S1[OR]) flag is set instead of the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the UART_S2[RXEDGIF] flag to set. The UART_S2[RXEDGIF] flag is cleared by writing a 1 to it. This function depends on the receiver being enabled (UART_C2[RE] = 1).

32.4.5 Baud rate tolerance

A transmitting device may operate at a baud rate below or above that of the receiver.

Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error will occur if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the RT8, RT9, and RT10 stop bit samples are a logic zero.

As the receiver samples an incoming frame, it re-synchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames will correct a misalignment between transmitter bit times and receiver bit times.

32.4.5.1 Slow data tolerance

Figure 32-20 shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

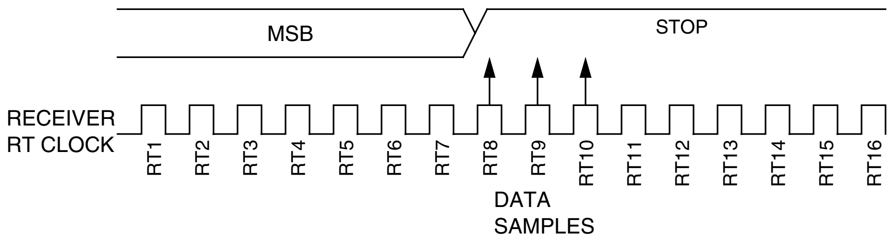


Figure 32-20. Slow data

For an 8-bit data and 1 stop bit character, data sampling of the stop bit takes the receiver 9 bit times x 16 RT cycles +10 RT cycles =154 RT cycles.

With the misaligned character shown in Figure 32-20, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit times x 16 RT cycles + 3 RT cycles = 147 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data and 1 stop bit character with no errors is:

((154 - 147) / 154) x 100 = 4.54%

For a 9-bit data or 2 stop bits character, data sampling of the stop bit takes the receiver 10 bit times x 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 32-20, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit times x 16 RT cycles + 3 RT cycles = 163 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit or 2 stop bits character with no errors is:

((170 - 163) / 170) X 100 = 4.12%

For a 9-bit data and 2 stop bit character, data sampling of the stop bit takes the receiver 11 bit times x 16 RT cycles + 10 RT cycles = 186 RT cycles.

With the misaligned character shown in Figure 32-20, the receiver counts 186 RT cycles at the point when the count of the transmitting device is 11 bit times x 16 RT cycles + 3 RT cycles = 179 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit and 2 stop bits character with no errors is: ((186 - 179) / 186) X 100 = 3.76%

32.4.5.2 Fast data tolerance

Figure 32-21 shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

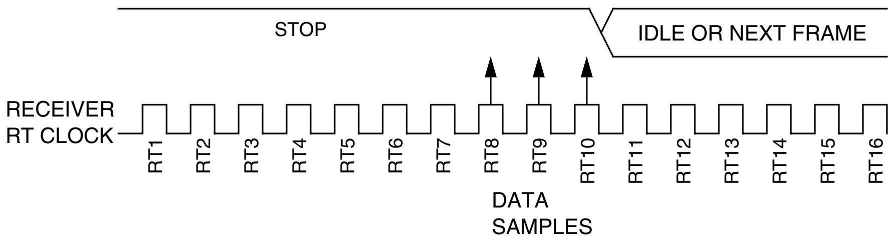


Figure 32-21. Fast data

For an 8-bit data and 1 stop bit character, data sampling of the stop bit takes the receiver 9 bit times x 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 32-21, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times x 16 RT cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit and 1 stop bit character with no errors is:

((154 - 160) / 154) x 100 = 3.90%

For a 9-bit data or 2 stop bits character, data sampling of the stop bit takes the receiver 10 bit times x 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times x 16 RT cycles = 176 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit or 2 stop bits character with no errors is:

((170 - 176) / 170) x 100 = 3.53%

For a 9-bit data and 2 stop bits character, data sampling of the stop bit takes the receiver 11 bit times x 16 RT cycles + 10 RT cycles = 186 RT cycles.

With the misaligned character shown in, the receiver counts 186 RT cycles at the point when the count of the transmitting device is 12 bit times x 16 RT cycles = 192 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit and 2 stop bits character with no errors is:

((186 - 192) / 186) x 100 = 3.23%

32.4.6 Additional UART functions

The following sections describe additional UART functions.

32.4.6.1 8- and 9-bit data modes

The UART system, transmitter and receiver, can be configured to operate in 9-bit data mode by setting UART_C1[M]. In 9-bit mode, there is a ninth data bit to the left of the most significant bit of the UART data register. For the transmit data buffer, this bit is stored in T8 in UART_C3. For the receiver, the ninth bit is held in UART_C3[R8].

For coherent writes to the transmit data buffer, write to UART_C3[T8] before writing to UART_D.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to UART_C3[T8] again. When data is transferred from the transmit data buffer to the transmit shifter, the value in UART_C3[T8] is copied at the same time data is transferred from UART_D to the shifter.

The 9-bit data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with address-mark wake-up so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

32.4.6.2 Stop mode operation

During all stop modes, clocks to the UART module are halted.

No UART module registers are affected in Stop mode.

The receive input active edge detect circuit remains active in Stop mode. An active edge on the receive input brings the CPU out of Stop mode if the interrupt is not masked (UART_BDH[RXEDGIE] = 1).

Because the clocks are halted, the UART module resumes operation upon exit from stop, only in Stop mode. Software must ensure stop mode is not entered while there is a character (including preamble, break and normal data) being transmitted out of or received into the UART module, that means UART_S1[TC] =1, UART_S1[TDRE] = 1, and UART_S2[RAF] = 0 must all meet before entering stop mode.

32.4.6.3 Loop mode

When UART_C1[LOOPS] is set, the UART_C1[RSRC] bit in the same register chooses between loop mode (UART_C1[RSRC] = 0) or single-wire mode (UART_C1[RSRC] = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.

32.4.6.4 Single-wire operation

When UART_C1[LOOPS] is set, UART_C1[RSRC] chooses between loop mode (UART_C1[RSRC] = 0) or single-wire mode (UART_C1[RSRC] = 1). Single-wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the UART_C3[TXDIR] bit controls the direction of serial data on the TxD pin. When UART_C3[TXDIR] is cleared, the TxD pin is an input to the UART receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When UART_C3[TXDIR] is set, the TxD pin is an output driven by the transmitter. In single-wire mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the UART, so it reverts to a general-purpose port I/O pin.

Chapter 33
General-Purpose Input/Output (GPIO)

33.1 Introduction

NOTE

For the chip-specific implementation details of this module's instances, see the chip configuration information.

The general-purpose input and output (GPIO) module is accessible via the peripheral bus and also communicates to the processor core via a zero wait state interface (IOPORT) for maximum pin performance. The GPIO registers support 8-bit, 16-bit or 32-bit accesses.

The GPIO data direction and output data registers control the direction and output data of each pin when the pin is configured for the GPIO function. The GPIO input data register displays the logic value on each pin when the pin is configured for any digital function, provided the corresponding Port Control and Interrupt module for that pin is enabled.

Efficient bit manipulation of the general-purpose outputs is supported through the addition of set, clear, and toggle write-only registers for each port output data register.

33.1.1 Features

- Features of the GPIO module include:
  - Port Data Input register visible in all digital pin-multiplexing modes
  - Port Data Input register with corresponding set/clear/toggle registers
  - Port Data Direction register
  - Zero wait state access to GPIO registers through IOPORT

NOTE

The GPIO module is clocked by system clock.

33.1.2 Modes of operation

The following table depicts different modes of operation and the behavior of the GPIO module in these modes.

Table 33-1. Modes of operation

Modes of operation	Description
Run	The GPIO module operates normally.
Wait	The GPIO module operates normally.
Stop	The GPIO module is disabled.
Debug	The GPIO module operates normally.

33.1.3 GPIO signal descriptions

Table 33-2. GPIO signal descriptions

GPIO signal descriptions	Description	I/O
PTA7–PTA0	General-purpose input/output	I/O
PTB7–PTB0	General-purpose input/output	I/O
PTC7–PTC0	General-purpose input/output	I/O

NOTE

Not all pins within each port are implemented on each device. See the chapter on signal multiplexing for the number of GPIO ports available in the device.

33.1.3.1 Detailed signal description

Table 33-3. GPIO interface-detailed signal descriptions

Signal	I/O	Description	
PTA7–PTA0 PTB7–PTB0 PTC7–PTC0	I/O	General-purpose input/output	
		State meaning	Asserted: The pin is logic 1. Deasserted: The pin is logic 0.
		Timing	Assertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time and input may be asserted asynchronously to the system clock.

Table 33-3. GPIO interface-detailed signal descriptions

Signal	I/O	Description	
			Deassertion: When output, this signal occurs on the rising-edge of the system clock. For input, it may occur at any time and input may be asserted asynchronously to the system clock.

33.2 Memory map and register definition

The GPIO module has two address slots on AIPS-Lite peripheral bridge to keep software compatibility between Freescale product portfolios. All of the GPIO registers could be accessed either by using base address of 0x400F_F000 or 0x4000_F000. It is recommended to use 0x400F_F000 as the base address of GPIO module, and the register memory map of this chapter is also based on the base address of 0x400F_F000.

Any read or write access to the GPIO memory space that is outside the valid memory map results in a bus error.

33.2.1 GPIO/FGPIO register bits assignment

In this device, each 8-bit port pin is mapped to the 32-bit GPIO/FGPIO registers as described in the table.

Table 33-4. GPIOA/FGPIOA register bits assignment

Register bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Port pin	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0

GPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_F000	Port Data Output Register (GPIOA_PDOR)	32	R/W	0000_0000h	33.2.2/590

Table continues on the next page...

GPIO memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_F004	Port Set Output Register (GPIOA_PSOR)	32	W (always reads 0)	0000_0000h	33.2.3/591
400F_F008	Port Clear Output Register (GPIOA_PCOR)	32	W (always reads 0)	0000_0000h	33.2.4/591
400F_F00C	Port Toggle Output Register (GPIOA_PTOR)	32	W (always reads 0)	0000_0000h	33.2.5/592
400F_F010	Port Data Input Register (GPIOA_PDIR)	32	R	0000_0000h	33.2.6/592
400F_F014	Port Data Direction Register (GPIOA_PDDR)	32	R/W	0000_0000h	33.2.7/593
400F_F018	Port Input Disable Register (GPIOA_PIDR)	32	R/W	FFFF_FFFFh	33.2.8/593

33.2.2 Port Data Output Register (GPIOx_PDOR)

This register configures the logic levels that are driven on each general-purpose output pins.

NOTE

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

Address: 400F_F000h base + 0h offset = 400F_F000h

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

GPIOx_PDOR field descriptions

Field	Description
PDO	Port Data Output Register bits for unbonded pins return a undefined value when read. 0 Logic level 0 is driven on pin, provided pin is configured for general-purpose output. 1 Logic level 1 is driven on pin, provided pin is configured for general-purpose output.

33.2.3 Port Set Output Register (GPIOx_PSOR)

This register configures whether to set the fields of the PDOR.

Address: 400F_F000h base + 4h offset = 400F_F004h

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

GPIOx_PSOR field descriptions

Field	Description
PTSO	Port Set Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to logic 1.

33.2.4 Port Clear Output Register (GPIOx_PCOR)

This register configures whether to clear the fields of PDOR.

Address: 400F_F000h base + 8h offset = 400F_F008h

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

GPIOx_PCOR field descriptions

Field	Description
PTCO	Port Clear Output Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is cleared to logic 0.

33.2.5 Port Toggle Output Register (GPIOx_PTOR)

Address: 400F_F000h base + Ch offset = 400F_F00Ch

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

GPIOx_PTOR field descriptions

Field	Description
PTTO	Port Toggle Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to the inverse of its existing logic state.

33.2.6 Port Data Input Register (GPIOx_PDIR)

NOTE

Do not modify pin configuration registers associated with pins not available in your selected package. All unbonded pins not available in your package will default to DISABLE state for lowest power consumption.

Address: 400F_F000h base + 10h offset = 400F_F010h

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

GPIOx_PDIR field descriptions

Field	Description
PDI	Port Data Input Reads 0 at the unimplemented pins for a particular device. Pins that are not configured for a digital function read 0. If the Port Control and Interrupt module is disabled, then the corresponding bit in PDIR does not update. 0 Pin logic level is logic 0, or is not configured for use by digital function. 1 Pin logic level is logic 1.

33.2.7 Port Data Direction Register (GPIOx_PDDR)

The PDDR configures the individual port pins for input or output.

Address: 400F_F000h base + 14h offset = 400F_F014h

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

GPIOx_PDDR field descriptions

Field	Description
PDD	Port Data Direction Configures individual port pins for input or output. 0 Pin is configured as general-purpose input, for the GPIO function. The pin will be high-Z if the port input is disabled in GPIOx_PIDR register. 1 Pin is configured as general-purpose output, for the GPIO function.

33.2.8 Port Input Disable Register (GPIOx_PIDR)

Address: 400F_F000h base + 18h offset = 400F_F018h

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

GPIOx_PIDR field descriptions

Field	Description
PID	Port Input Disable 0 Pin is configured for General Purpose Input, provided the pin is configured for any digital function. 1 Pin is not configured as General Purpose Input.Corresponding Port Data Input Register bit will read zero.

33.3 FGPIO memory map and register definition

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800_0000.

Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle. This aliased Fast GPIO memory map is called FGPIO.

Any read or write access to the FGPIO memory space that is outside the valid memory map results in a bus error. All register accesses complete with zero wait states, except error accesses which complete with one wait state.

33.3.1 GPIO/FGPIO register bits assignment

In this device, each 8-bit port pin is mapped to the 32-bit GPIO/FGPIO registers as described in the table.

Table 33-21. GPIOA/FGPIOA register bits assignment

Register bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Port pin	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0

FGPIO memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F800_0000	Port Data Output Register (FGPIOA_PDOR)	32	R/W	0000_0000h	33.3.2/595
F800_0004	Port Set Output Register (FGPIOA_PSOR)	32	W (always reads 0)	0000_0000h	33.3.3/595
F800_0008	Port Clear Output Register (FGPIOA_PCOR)	32	W (always reads 0)	0000_0000h	33.3.4/596
F800_000C	Port Toggle Output Register (FGPIOA_PTOR)	32	W (always reads 0)	0000_0000h	33.3.5/596
F800_0010	Port Data Input Register (FGPIOA_PDIR)	32	R	0000_0000h	33.3.6/597
F800_0014	Port Data Direction Register (FGPIOA_PDDR)	32	R/W	0000_0000h	33.3.7/597
F800_0018	Port Input Disable Register (FGPIOA_PIDR)	32	R/W	FFFF_FFFFh	33.3.8/598

33.3.2 Port Data Output Register (FGPIOx_PDOR)

This register configures the logic levels that are driven on each general-purpose output pins.

Address: F800_0000h base + 0h offset = F800_0000h

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

FGPIOx_PDOR field descriptions

Field	Description
PDO	Port Data Output Unimplemented pins for a particular device read as zero. 0 Logic level 0 is driven on pin, provided pin is configured for general-purpose output. 1 Logic level 1 is driven on pin, provided pin is configured for general-purpose output.

33.3.3 Port Set Output Register (FGPIOx_PSOR)

This register configures whether to set the fields of the PDOR.

Address: F800_0000h base + 4h offset = F800_0004h

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

FGPIOx_PSOR field descriptions

Field	Description
PTSO	Port Set Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to logic 1.

33.3.4 Port Clear Output Register (FGPIOx_PCOR)

This register configures whether to clear the fields of PDOR.

Address: F800_0000h base + 8h offset = F800_0008h

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

FGPIOx_PCOR field descriptions

Field	Description
PTCO	Port Clear Output Writing to this register will update the contents of the corresponding bit in the Port Data Output Register (PDOR) as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is cleared to logic 0.

33.3.5 Port Toggle Output Register (FGPIOx_PTOR)

Address: F800_0000h base + Ch offset = F800_000Ch

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

FGPIOx_PTOR field descriptions

Field	Description
PTTO	Port Toggle Output Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0 Corresponding bit in PDORn does not change. 1 Corresponding bit in PDORn is set to the inverse of its existing logic state.

33.3.6 Port Data Input Register (FGPIOx_PDIR)

Address: F800_0000h base + 10h offset = F800_0010h

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

FGPIOx_PDIR field descriptions

Field	Description
PDI	Port Data Input Reads 0 at the unimplemented pins for a particular device. Pins that are not configured for a digital function read 0. If the Port Control and Interrupt module is disabled, then the corresponding bit in PDIR does not update. 0 Pin logic level is logic 0, or is not configured for use by digital function. 1 Pin logic level is logic 1.

33.3.7 Port Data Direction Register (FGPIOx_PDDR)

The PDDR configures the individual port pins for input or output.

Address: F800_0000h base + 14h offset = F800_0014h

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

FGPIOx_PDDR field descriptions

Field	Description
PDD	Port Data Direction Configures individual port pins for input or output. 0 Pin is configured as general-purpose input, for the GPIO function. The pin will be high-Z if the port input is disabled in FPIOx_PIDR register. 1 Pin is configured as general-purpose output, for the GPIO function.

33.3.8 Port Input Disable Register (FGPIOx_PIDR)

Address: F800_0000h base + 18h offset = F800_0018h

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

FGPIOx_PIDR field descriptions

Field	Description
PID	Port Input Disable 0 Pin is configured for General Purpose Input, provided the pin is configured for any digital function. 1 Pin is not configured as General Purpose Input. Corresponding Port Data Input Register bit will read zero.

33.4 Functional description

33.4.1 General-purpose input

The logic state of each pin is available via the Port Data Input registers, provided the pin is configured for a digital function and the corresponding Port Control and Interrupt module is enabled.

33.4.2 General-purpose output

The logic state of each pin can be controlled via the port data output registers and port data direction registers, provided the pin is configured for the GPIO function. The following table depicts the conditions for a pin to be configured as input/output.

If	Then
A pin is configured for the GPIO function and the corresponding port data direction register bit is clear.	The pin is configured as an input.
A pin is configured for the GPIO function and the corresponding port data direction register bit is set.	The pin is configured as an output and and the logic state of the pin is equal to the corresponding port data output register.

To facilitate efficient bit manipulation on the general-purpose outputs, pin data set, pin data clear, and pin data toggle registers exist to allow one or more outputs within one port to be set, cleared, or toggled from a single register write.

The corresponding Port Control and Interrupt module does not need to be enabled to update the state of the port data direction registers and port data output registers including the set/clear/toggle registers.

33.4.3 IOPORT

The GPIO registers are also aliased to the IOPORT interface on the Cortex-M0+ from address 0xF800_0000. Accesses via the IOPORT interface occur in parallel with any instruction fetches and will therefore complete in a single cycle.

Chapter 34
Keyboard Interrupts (KBI)

34.1 Introduction

34.1.1 Features

The KBI features include:

- Up to eight keyboard interrupt pins with individual pin enable bits
- Each keyboard interrupt pin is programmable as:
  - falling-edge sensitivity only
  - rising-edge sensitivity only
  - both falling-edge and low-level sensitivity
  - both rising-edge and high-level sensitivity
- One software-enabled keyboard interrupt
- Exit from low-power modes

34.1.2 Modes of Operation

This section defines the KBI operation in:

- Wait mode
- Stop mode
- Background debug mode

34.1.2.1 KBI in Wait mode

Executing the Wait instruction places the MCU into Wait mode. The KBI interrupt should be enabled (KBI_SC[KBIE] = 1), if desired, before executing the Wait instruction, allowing the KBI to continue to operate while the MCU is in Wait mode. An enabled KBI pin (KBI_PE[KBIPEn] = 1) can be used to bring the MCU out of Wait mode if the KBI interrupt is enabled (KBI_SC[KBIE] = 1).

34.1.2.2 KBI in Stop modes

Executing the Stop instruction places the MCU into Stop mode (when Stop is selected), where the KBI can operate asynchronously. If this is the desired behavior, the KBI interrupt must be enabled (KBI_SC[KBIE] = 1) before executing the Stop instruction, allowing the KBI to continue to operate while the MCU is in Stop mode. An enabled KBI pin (KBI_PE[KBIPEn] = 1) can be used to bring the MCU out of Stop mode if the KBI interrupt is enabled (KBI_SC[KBIE] = 1).

34.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown below..

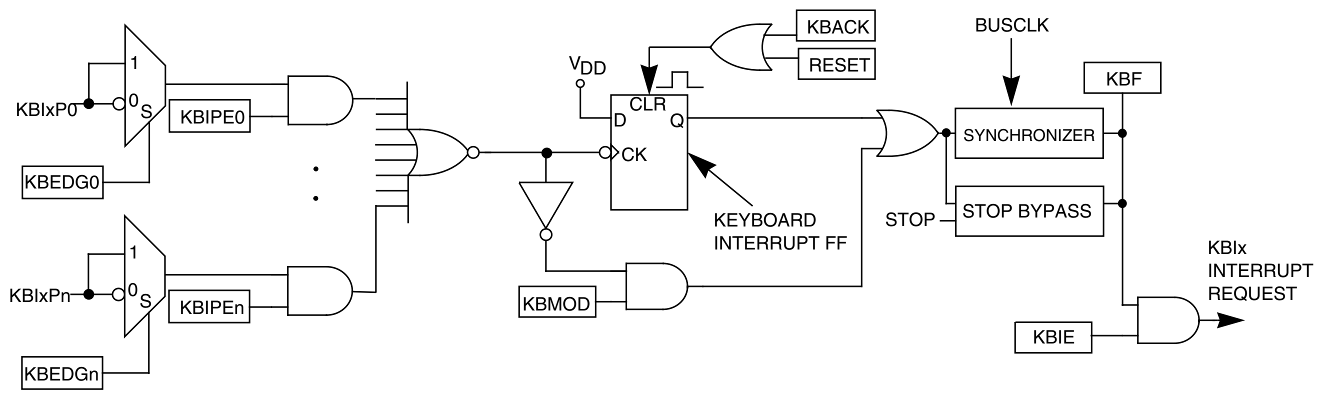


Figure 34-1. KBI block diagram

34.2 External signals description

The KBI input pins can be used to detect either falling edges, or both falling edge and low-level interrupt requests. The KBI input pins can also be used to detect either rising edges, or both rising edge and high-level interrupt requests.

The signal properties of KBI are shown in the following table:

Table 34-1. External signals description

Signal	Function	I/O
KBIxPn	Keyboard interrupt pins	I

34.3 Register definition

The KBI includes following registers:

- A pin status and control register, KBIx_SC
- A pin enable register, KBIx_PE
- An edge select register, KBIx_ES

See the direct-page register summary in the Memory chapter for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names.

Some MCUs may have more than one KBI, so register names include placeholder characters to identify which KBI is being referenced.

34.4 Memory Map and Registers

KBI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_9000	KBI Status and Control Register (KBI0_SC)	8	R/W	00h	34.4.1/604
4007_9001	KBIx Pin Enable Register (KBI0_PE)	8	R/W	00h	34.4.2/604
4007_9002	KBIx Edge Select Register (KBI0_ES)	8	R/W	00h	34.4.3/605
4007_A000	KBI Status and Control Register (KBI1_SC)	8	R/W	00h	34.4.1/604
4007_A001	KBIx Pin Enable Register (KBI1_PE)	8	R/W	00h	34.4.2/604
4007_A002	KBIx Edge Select Register (KBI1_ES)	8	R/W	00h	34.4.3/605

34.4.1 KBI Status and Control Register (KBIx_SC)

KBI_SC contains the status flag and control bits, which are used to configure the KBI.

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read	0				KBF	▨	KBIE	KBMOD
Write	▨				▨	KBACK		
Reset	0	0	0	0	0	0	0	0

KBIx_SC field descriptions

Field	Description
7–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 KBF	KBI Interrupt Flag KBF indicates when a KBI interrupt request is detected. Writes have no effect on KBF. 0 KBI interrupt request not detected. 1 KBI interrupt request detected.
2 KBACK	KBI Acknowledge Writing a 1 to KBACK is part of the flag clearing mechanism.
1 KBIE	KBI Interrupt Enable KBIE determines whether a KBI interrupt is enabled or not. 0 KBI interrupt not enabled. 1 KBI interrupt enabled.
0 KBMOD	KBI Detection Mode KBMOD (along with the KBEDG bits) controls the detection mode of the KBI interrupt pins. 0 Keyboard detects edges only. 1 Keyboard detects both edges and levels.

34.4.2 KBIx Pin Enable Register (KBIx_PE)

KBIx_PE contains the pin enable control bits.

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read Write	KBIPE							
Reset	0	0	0	0	0	0	0	0

KBIx_PE field descriptions

Field	Description
KBIPE	KBI Pin Enables Each of the KBIPEn bits enable the corresponding KBI interrupt pin. 0 Pin is not enabled as KBI interrupt. 1 Pin is enabled as KBI interrupt.

34.4.3 KBIx Edge Select Register (KBIx_ES)

KBIx_ES contains the edge select control bits.

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read Write	KBEDG							
Reset	0	0	0	0	0	0	0	0

KBIx_ES field descriptions

Field	Description
KBEDG	KBI Edge Selects Each of the KBEDGn bits selects the falling edge/low-level or rising edge/high-level function of the corresponding pin. 0 Falling edge/low level. 1 Rising edge/high level.

34.5 Functional Description

This on-chip peripheral module is called a keyboard interrupt module because originally it was designed to simplify the connection and use of row-column matrices of keyboard switches. However, these inputs are also useful as extra external interrupt inputs and as an external means of waking the MCU from stop or wait low-power modes.

The KBI module allows up to eight pins to act as additional interrupt sources. Writing to the KBIx_PE[KBIPEn] bits independently enables or disables each KBI pin. Each KBI pin can be configured as edge sensitive or edge and level sensitive based on the KBIx_SC[KBMOD] bit. Edge sensitive can be software programmed to be either falling or rising; the level can be either low or high. The polarity of the edge or edge and level sensitivity is selected using the KBIx_ES[KBEDGn] bits.

34.5.1 Edge-only sensitivity

Synchronous logic is used to detect edges. A falling edge is detected when an enabled keyboard interrupt (KBIx_PE[KBIPEn]=1) input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input signal is seen as a logic 0 (the deasserted level) during one bus cycle and then a logic 1 (the asserted level) during the next cycle.

Before the first edge is detected, all enabled keyboard interrupt input signals must be at the deasserted logic levels. After any edge is detected, all enabled keyboard interrupt input signals must return to the deasserted level before any new edge can be detected.

A valid edge on an enabled KBI pin will set KBIx_SC[KBF]. If KBIx_SC[KBIE] is set, an interrupt request will be presented to the MPU. Clearing of KBIx_SC[KBF] is accomplished by writing a 1 to KBIx_SC[KBACK].

34.5.2 Edge and level sensitivity

A valid edge or level on an enabled KBI pin will set KBIx_SC[KBF]. If KBIx_SC[KBIE] is set, an interrupt request will be presented to the MCU. Clearing of KBIx_SC[KBF] is accomplished by writing a 1 to KBIx_SC[KBACK], provided all enabled keyboard inputs are at their deasserted levels. KBIx_SC[KBF] will remain set if any enabled KBI pin is asserted while attempting to clear KBIx_SC[KBF] by writing a 1 to KBIx_SC[KBACK].

34.5.3 KBI Pullup Resistor

Each KBI pin, if enabled by KBIx_PE, can be configured via the associated I/O port pull enable register, see Port Control (PORT) chapter, to use:

- an internal pullup resistor, or
- no resistor

If an internal pullup resistor is enabled for an enabled KBI pin, the associated I/O port pull select register (see I/O Port chapter) can be used to select an internal pullup resistor.

34.5.4 KBI initialization

When a keyboard interrupt pin is first enabled, it is possible to get a false keyboard interrupt flag. To prevent a false interrupt request during keyboard initialization, the user should do the following:

1. Mask keyboard interrupts by clearing KBIx_SC[KBIE].
2. Enable the KBI polarity by setting the appropriate KBIx_ES[KBEDGn] bits.
3. Before using internal pullup resistors, configure the associated bits in PORT_PUEL and PORT_PUEH.
4. Enable the KBI pins by setting the appropriate KBIx_PE[KBIPEn] bits.
5. Write to KBIx_SC[KBACK] to clear any false interrupts.
6. Set KBIx_SC[KBIE] to enable interrupts.

Appendix A
Revision history

A.1 Changes between revisions 3 and 2

Table A-1. Changes between revisions 3 and 2

Chapter	Description
Chip Configuration	• Updated System interconnection diagram.
Clock Distribution	• Updated a note in section of FTM and PWT clocking section to "And the on-chip timer clock (TIMER_CLK) must be at least 4x faster than the external clock from TCLK1 or TCLK2."
Flash Memory Module (FTMRE)	• Edited the chapter to clear typo errors. • Updated to "If an is detected during the reset sequence, both FSTAT[MGSTAT] bits will be set." in the Flash initialization after system reset section. • Updated Flash command summary section. • Removed Configure NVM command
Analog-to-digital converter (ADC)	• Updated FIFO operation to support hardware trigger multiple conversion feature
Pulse Width Timer (PWT)	• Updated the figure of "Trigger edge detection and pulse width registers update".

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number MKE04P24M48SF0RM
Revision 3, Feburary 2014